

ELE HSM API Rev 1.0

NXP Copyright

Generated by Doxygen 1.8.17

1 ELE HSM API	1
2 Revision History	1
3 General concepts related to the API	2
3.1 Session	2
3.2 Service flow	2
3.3 Example	3
3.4 Key store	3
3.4.1 Key management	3
3.4.2 NVM writing	4
3.5 Implementation specificities	4
4 Module Index	4
4.1 Modules	4
5 Module Documentation	5
5.1 Session	5
5.1.1 Detailed Description	6
5.1.2 Data Structure Documentation	6
5.1.3 Typedef Documentation	6
5.1.4 Function Documentation	7
5.2 Key management	10
5.2.1 Detailed Description	12
5.2.2 Data Structure Documentation	12
5.2.3 Macro Definition Documentation	15
5.2.4 Typedef Documentation	16
5.2.5 Enumeration Type Documentation	17
5.2.6 Function Documentation	18
5.3 Cipherring	22
5.3.1 Detailed Description	22
5.3.2 Data Structure Documentation	22
5.3.3 Macro Definition Documentation	23
5.3.4 Typedef Documentation	24
5.3.5 Enumeration Type Documentation	24
5.3.6 Function Documentation	25
5.4 Signature generation	28
5.4.1 Detailed Description	29
5.4.2 Data Structure Documentation	29
5.4.3 Macro Definition Documentation	29
5.4.4 Typedef Documentation	30
5.4.5 Enumeration Type Documentation	30
5.4.6 Function Documentation	30
5.5 Signature verification	34

5.5.1 Detailed Description	34
5.5.2 Data Structure Documentation	34
5.5.3 Macro Definition Documentation	35
5.5.4 Typedef Documentation	36
5.5.5 Function Documentation	36
5.6 Random number generation	39
5.6.1 Detailed Description	39
5.6.2 Data Structure Documentation	39
5.6.3 Function Documentation	39
5.7 Hashing	41
5.7.1 Detailed Description	41
5.7.2 Data Structure Documentation	41
5.7.3 Macro Definition Documentation	42
5.7.4 Enumeration Type Documentation	42
5.7.5 Function Documentation	42
5.8 Data storage	44
5.8.1 Detailed Description	44
5.8.2 Data Structure Documentation	44
5.8.3 Typedef Documentation	45
5.8.4 Function Documentation	45
5.9 Authenticated Encryption	48
5.9.1 Detailed Description	48
5.9.2 Function Documentation	48
5.10 Mac	49
5.10.1 Detailed Description	49
5.10.2 Data Structure Documentation	49
5.10.3 Macro Definition Documentation	50
5.10.4 Typedef Documentation	50
5.10.5 Function Documentation	51
5.11 Dump Firmware Log	54
5.11.1 Detailed Description	54
5.11.2 Data Structure Documentation	54
5.11.3 Function Documentation	54
5.12 Dev attest	55
5.12.1 Detailed Description	55
5.12.2 Data Structure Documentation	55
5.12.3 Function Documentation	55
5.13 Dev Info	57
5.13.1 Detailed Description	57
5.13.2 Data Structure Documentation	57
5.13.3 Function Documentation	57
5.14 Generic Crypto: Asymmetric Crypto	59

5.14.1 Detailed Description	60
5.14.2 Data Structure Documentation	60
5.14.3 Macro Definition Documentation	61
5.14.4 Typedef Documentation	61
5.14.5 Enumeration Type Documentation	62
5.14.6 Function Documentation	62
5.15 Generic Crypto Asymmetric Key Generate	63
5.15.1 Detailed Description	63
5.15.2 Data Structure Documentation	63
5.15.3 Function Documentation	63
5.16 Get Info	65
5.16.1 Detailed Description	65
5.16.2 Data Structure Documentation	65
5.16.3 Function Documentation	65
5.17 Public key recovery	67
5.17.1 Detailed Description	67
5.17.2 Data Structure Documentation	67
5.17.3 Function Documentation	67
5.18 Key store	69
5.18.1 Detailed Description	69
5.18.2 Data Structure Documentation	69
5.18.3 Macro Definition Documentation	70
5.18.4 Typedef Documentation	70
5.18.5 Function Documentation	70
5.19 Life Cycle update	72
5.19.1 Detailed Description	72
5.19.2 Data Structure Documentation	72
5.19.3 Enumeration Type Documentation	72
5.19.4 Function Documentation	72
5.20 Error codes	74
5.20.1 Detailed Description	74
5.20.2 Enumeration Type Documentation	74
Index	77

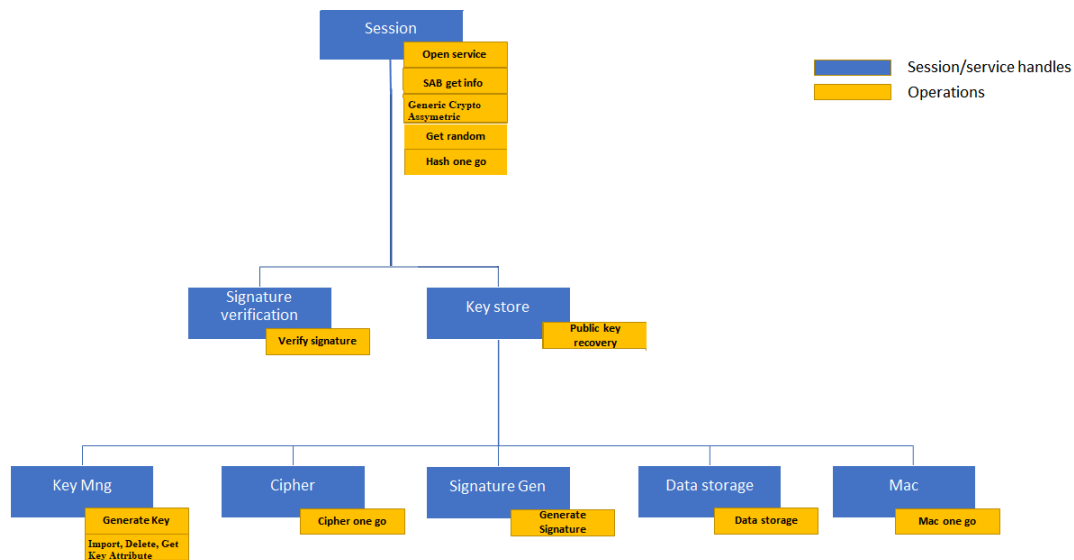
1 ELE HSM API

This document is a software referece description of the API provided by the i.MX8ULP, i.MX93 HSM solutions for ELE Platform.

2 Revision History

Revision	date	description
0.1	Apr 27 2023	Preliminary draft

3 General concepts related to the API



3.1 Session

The API must be initialized by a potential requestor by opening a session.

The session establishes a route (MU, DomainID...) between the requestor and the HSM. When a session is opened, the HSM returns a handle identifying the session to the requestor.

3.2 Service flow

For a given category of services which require service handle, the requestor is expected to open a service flow by invoking the appropriate HSM API.

The session handle, as well as the control data needed for the service flow, are provided as parameters of the call. Upon reception of the open request, the HSM allocates a context in which the session handle, as well as the provided control parameters are stored and return a handle identifying the service flow.

The context is preserved until the service flow, or the session, are closed by the user and it is used by the HSM to proceed with the sub-subsequent operations requested by the user on the service flow.

3.3 Example

```
/* Open a session: create a route between the user and the HSM */
hsm_open_session(&open_session_args, &session_hdl);

/* Open a key store - user is authenticated */
hsm_open_key_store_service(session_hdl, &open_svc_key_store_args, &key_store_hdl);

/* Open cipher service - it grants access to ciphering operations */
hsm_open_cipher_service(key_store_hdl, &open_svc_cipher_args, &cipher_hdl);

/* Perform ECB, CCB ... */
hsm_cipher_one_go (cipher_hdl, &op_cipher_one_go_args);
/* Perform authenticate and encryption algos: e.g GCM */
hsm_auth_enc (cipher_hdl, &op_auth_enc_args);
/* Perform hashing operations: e.g SHA */
hsm_hash_one_go (hash_hdl, &op_hash_one_go_args);

/* Close the session and all the related services */
hsm_close_session(session_hdl);
```

3.4 Key store

A key store can be created by specifying the CREATE flag in the hsm_open_key_store_service API. Please note that the created key store will be not stored in the NVM till a key is generated or imported specifying the "STRICT OPERATION" flag.

Only symmetric and private keys are stored into the key store. Public keys can be exported during the key pair generation operation or recalculated through the hsm_pub_key_recovery API.

Secret keys cannot be exported under any circumstances, while they can be imported in encrypted form.

3.4.1 Key management

Keys are divided in groups, keys belonging to the same group are written/read from the NVM as a monolithic block. Up to 3 key groups can be handled in the HSM local memory (those immediately available to perform crypto operations), while up to 1000 key groups can be handled in the external NVM and imported in the local memory as needed.

If the local memory is full (3 key groups already reside in the HSM local memory) and a new key group is needed by an incoming user request, the HSM swaps one of the local key group with the one needed by the user request. The user can control which key group must be kept in the local memory (cached) through the manage_key_group API lock/unlock mechanism.

As general concept, frequently used keys should be kept, when possible, in the same key group and locked in the local memory for performance optimization.

3.4.2 NVM writing

All the APIs creating a key store (open key store API) or modifying its content (key generation, key_management, key derivation functions) provide a "STRICT OPERATION" flag. If the flag is set, the HSM exports the relevant key store blocks into the external NVM and increments (blows one bit) the OTP monotonic counter used as roll back protection. In case of key generation/derivation /update the "STRICT OPERATION" has effect only on the target key group.

Any update to the key store must be considered as effective only after an operation specifying flag "STRICT OPERATION" is acknowledged by the HSM. All the operations not specifying the "STRICT OPERATION" flags impact the HSM local memory only and will be lost in case of system reset

Due to the limited monotonic counter size, the user should, when possible, perform multiple update before setting the "STRICT OPERATION" flag(i.e. keys to be updated should be kept in the same key group).

Once the monotonic counter is completely blown a warning is returned on each key store export to the NVM to inform the user that the new updates are not roll-back protected.

3.5 Implementation specificities

HSM API with common features are supported on i.MX8ULP and i.MX93. The details of supported features per chip will be listed in the platform specificities.

4 Module Index

4.1 Modules

Here is a list of all modules:

Session	5
Key management	10
Ciphering	22
Signature generation	28
Signature verification	34
Random number generation	39
Hashing	41
Data storage	44
Authenticated Encryption	48
Mac	49
Dump Firmware Log	54
Dev attest	55
Dev Info	57
Generic Crypto: Asymmetric Crypto	59
Generic Crypto Asymmetric Key Generate	63

Get Info	65
Public key recovery	67
Key store	69
Life Cycle update	72
Error codes	74

5 Module Documentation

5.1 Session

The API must be initialized by a potential requestor by opening a session. Once a session is closed all the associated service flows are closed by the HSM.

Data Structures

- struct [hsm_session_hdl_s](#)
- struct [hsm_service_hdl_s](#)
- struct [open_session_args_t](#)

Macros

- #define [HSM_MAX_SESSIONS](#) (8u)
Maximum sessions supported.
- #define [HSM_MAX_SERVICES](#) (32u)
Maximum services supported.
- #define [HSM_OPEN_SESSION_PRIORITY_LOW](#) (0x00U)
Low priority. default setting on platforms that doesn't support sessions priorities.
- #define [HSM_OPEN_SESSION_PRIORITY_HIGH](#) (0x01U)
High Priority session.
- #define [HSM_OPEN_SESSION_FIPS_MODE_MASK](#) (1u << 0)
Only FIPS certified operations authorized in this session.
- #define [HSM_OPEN_SESSION_EXCLUSIVE_MASK](#) (1u << 1)
No other HSM session will be authorized on the same security enclave.
- #define [HSM_OPEN_SESSION_LOW_LATENCY_MASK](#) (1u << 3)
Use a low latency HSM implementation.
- #define [HSM_OPEN_SESSION_NO_KEY_STORE_MASK](#) (1u << 4)
No key store will be attached to this session. May provide better performances on some operation depending on the implementation. Usage of the session will be restricted to operations that doesn't involve secret keys (e.g. hash, signature verification, random generation).
- #define [HSM_OPEN_SESSION_RESERVED_MASK](#) ((1u << 2) | (1u << 5) | (1u << 6) | (1u << 7))
Bits reserved for future use. Should be set to 0.

Typedefs

- typedef uint32_t [hsm_hdl_t](#)

Functions

- `hsm_err_t hsm_open_session (open_session_args_t *args, hsm_hdl_t *session_hdl)`
- `hsm_err_t hsm_close_session (hsm_hdl_t session_hdl)`
- `struct hsm_session_hdl_s * session_hdl_to_ptr (uint32_t hdl)`
- `struct hsm_service_hdl_s * service_hdl_to_ptr (uint32_t hdl)`
- `void delete_session (struct hsm_session_hdl_s *s_ptr)`
- `void delete_service (struct hsm_service_hdl_s *s_ptr)`
- `struct hsm_session_hdl_s * add_session (void)`
- `struct hsm_service_hdl_s * add_service (struct hsm_session_hdl_s *session)`

5.1.1 Detailed Description

The API must be initialized by a potential requestor by opening a session. Once a session is closed all the associated service flows are closed by the HSM.

5.1.2 Data Structure Documentation

5.1.2.1 `struct hsm_session_hdl_s` Structure describing the session handle members

Data Fields

<code>struct plat_os_abs_hdl *</code>	<code>phdl</code>	Pointer to OS device node.
<code>uint32_t</code>	<code>session_hdl</code>	Session handle.
<code>uint32_t</code>	<code>mu_type</code>	Session MU type.

5.1.2.2 `struct hsm_service_hdl_s` Structure describing the service handle members

Data Fields

<code>struct hsm_session_hdl_s *</code>	<code>session</code>	Pointer to session handle.
<code>uint32_t</code>	<code>service_hdl</code>	Service handle.

5.1.2.3 `struct open_session_args_t` Structure detailing the open session operation member arguments

Data Fields

<code>uint32_t</code>	<code>session_hdl</code>	Session handle.
<code>uint8_t</code>	<code>session_priority</code>	Priority of the operations performed in this session.
<code>uint8_t</code>	<code>operating_mode</code>	Options for the session to be opened (bitfield).
<code>uint8_t</code>	<code>interrupt_idx</code>	Interrupt number of the MU used to indicate data availability.

5.1.3 Typedef Documentation

5.1.3.1 hsm_hdl_t

```
typedef uint32_t hsm_hdl_t
```

Define the HSM handle type

5.1.4 Function Documentation

5.1.4.1 hsm_open_session()

```
hsm_err_t hsm_open_session (
    open_session_args_t * args,
    hsm_hdl_t * session_hdl )
```

Parameters

<i>args</i>	pointer to the structure containing the function arguments.
<i>session_hdl</i>	pointer to where the session handle must be written.

Returns

error_code error code.

5.1.4.2 hsm_close_session()

```
hsm_err_t hsm_close_session (
    hsm_hdl_t session_hdl )
```

Terminate a previously opened session. All the services opened under this session are closed as well

Parameters

<i>session_hdl</i>	pointer to the handle identifying the session to be closed.
--------------------	---

Returns

error_code error code.

5.1.4.3 session_hdl_to_ptr()

```
struct hsm_session_hdl_s* session_hdl_to_ptr (
    uint32_t hdl )
```

Returns pointer to the session handle

Parameters

<i>hdl</i>	identifying the session handle.
------------	---------------------------------

Returns

pointer to the session handle.

5.1.4.4 service_hdl_to_ptr() `struct hsm_service_hdl_s* service_hdl_to_ptr (`
`uint32_t hdl)`

Returns pointer to the service handle

Parameters

<i>hdl</i>	identifying the session handle.
------------	---------------------------------

Returns

pointer to the service handle.

5.1.4.5 delete_session() `void delete_session (`
`struct hsm_session_hdl_s * s_ptr)`

Delete the session

Parameters

<i>s_ptr</i>	pointer identifying the session.
--------------	----------------------------------

5.1.4.6 delete_service() `void delete_service (`
`struct hsm_service_hdl_s * s_ptr)`

Delete the service

Parameters

<i>s_ptr</i>	pointer identifying the service.
--------------	----------------------------------

5.1.4.7 add_session() `struct hsm_session_hdl_s* add_session (`
`void)`

Add the session

Returns

pointer to the session.

5.1.4.8 add_service() `struct hsm_service_hdl_s* add_service (`
`struct hsm_session_hdl_s * session)`

Add the service

Returns

pointer to the service.

5.2 Key management

Data Structures

- struct [op_delete_key_args_t](#)
- struct [op_get_key_attr_args_t](#)
- struct [op_import_key_args_t](#)
- struct [kek_enc_key_hdr_t](#)
- struct [op_generate_key_ext_args_t](#)
- struct [op_generate_key_args_t](#)
- struct [open_svc_key_management_args_t](#)

Macros

- #define [HSM_OP_DEL_KEY_FLAGS_STRICT_OPERATION](#) ((hsm_op_import_key_flags_t)(1u << 7))
- #define [HSM_OP_IMPORT_KEY_INPUT_E2GO_TLV](#) ((hsm_op_import_key_flags_t)(1u << 0))
Bit 0: set 1 means input is E2GO_TLV.
- #define [HSM_OP_IMPORT_KEY_INPUT_SIGNED_MSG](#) ((hsm_op_import_key_flags_t)(0u << 0))
Bit 0: set 0 means input is signed message.
- #define [HSM_OP_IMPORT_KEY_FLAGS_STRICT_OPERATION](#) ((hsm_op_import_key_flags_t)(1u << 7))
Bit 7: Strict: Request completed - New key written to NVM with updated MC.
- #define [HSM_KEY_USAGE_EXPORT](#) ((hsm_key_usage_t) (1u << 0))
- #define [HSM_KEY_USAGE_ENCRYPT](#) ((hsm_key_usage_t) (1u << 8))
- #define [HSM_KEY_USAGE_DECRYPT](#) ((hsm_key_usage_t) (1u << 9))
- #define [HSM_KEY_USAGE_SIGN_MSG](#) ((hsm_key_usage_t) (1u << 10))
- #define [HSM_KEY_USAGE_VERIFY_MSG](#) ((hsm_key_usage_t) (1u << 11))
- #define [HSM_KEY_USAGE_SIGN_HASH](#) ((hsm_key_usage_t) (1u << 12))
- #define [HSM_KEY_USAGE_VERIFY_HASH](#) ((hsm_key_usage_t) (1u << 13))
- #define [HSM_KEY_USAGE_DERIVE](#) ((hsm_key_usage_t) (1u << 14))
- #define [HSM_KEY_INFO_PERSISTENT](#) ((hsm_key_info_t)(0u << 1))
- #define [HSM_KEY_INFO_PERMANENT](#) ((hsm_key_info_t)(1u << 0))
- #define [HSM_KEY_INFO_TRANSIENT](#) ((hsm_key_info_t)(1u << 1))
- #define [HSM_KEY_INFO_MASTER](#) ((hsm_key_info_t)(1u << 2))
- #define [HSM_KEY_INFO_KEK](#) ((hsm_key_info_t)(1u << 3))
- #define [FLAG_0](#)
- #define [HSM_OP_KEY_GENERATION_FLAGS_STRICT_OPERATION](#) ((hsm_op_key_gen_flags_t)(1u << 7))

Typedefs

- typedef uint8_t [hsm_op_delete_key_flags_t](#)
- typedef uint8_t [hsm_op_import_key_flags_t](#)
- typedef uint32_t [hsm_key_usage_t](#)
- typedef uint16_t [hsm_key_group_t](#)
- typedef uint16_t [hsm_key_info_t](#)
- typedef uint8_t [hsm_op_key_gen_flags_t](#)
- *Reserverd Bits 0 - 6.*
- typedef uint8_t [hsm_svc_key_management_flags_t](#)

Enumerations

- enum `hsm_storage_loc_t` { `HSM_SE_KEY_STORAGE` = 0x00000000 }
- enum `hsm_storage_persist_lvl_t` {
`HSM_VOLATILE_STORAGE` = 0x0,
`HSM_PERSISTENT_STORAGE` = 0x1,
`HSM_PERMANENT_STORAGE` = 0xFF }
- enum `hsm_key_lifetime_t` {
`HSM_SE_KEY_STORAGE_VOLATILE` = `HSM_SE_KEY_STORAGE` | `HSM_VOLATILE_STORAGE`,
`HSM_SE_KEY_STORAGE_PERSISTENT` = `HSM_SE_KEY_STORAGE` | `HSM_PERSISTENT_STORAGE`,
`HSM_SE_KEY_STORAGE_PERS_PERM` = `HSM_SE_KEY_STORAGE` | `HSM_PERMANENT_STORAGE`
}
- enum `hsm_pubkey_type_t` {
`HSM_PUBKEY_TYPE_RSA` = 0x4001,
`HSM_PUBKEY_TYPE_ECC_BP_R1` = 0x4130,
`HSM_PUBKEY_TYPE_ECC_NIST` = 0x4112,
`HSM_PUBKEY_TYPE_ECC_BP_T1` = 0xC180 }
- enum `hsm_key_type_t` {
`HSM_KEY_TYPE_HMAC` = 0x1100,
`HSM_KEY_TYPE_AES` = 0x2400,
`HSM_KEY_TYPE_SM4` = 0x2405,
`HSM_KEY_TYPE_RSA` = 0x7001,
`HSM_KEY_TYPE_ECC_BP_R1` = 0x7130,
`HSM_KEY_TYPE_ECC_NIST` = 0x7112 }
- enum `hsm_bit_key_sz_t` {
`HSM_KEY_SIZE_HMAC_224` = 224,
`HSM_KEY_SIZE_HMAC_256` = 256,
`HSM_KEY_SIZE_HMAC_384` = 384,
`HSM_KEY_SIZE_HMAC_512` = 512,
`HSM_KEY_SIZE_AES_128` = 128,
`HSM_KEY_SIZE_AES_192` = 192,
`HSM_KEY_SIZE_AES_256` = 256,
`HSM_KEY_SIZE_SM4_128` = 128,
`HSM_KEY_SIZE_RSA_2048` = 2048,
`HSM_KEY_SIZE_RSA_3072` = 3072,
`HSM_KEY_SIZE_RSA_4096` = 4096,
`HSM_KEY_SIZE_ECC_BP_R1_224` = 224,
`HSM_KEY_SIZE_ECC_BP_R1_256` = 256,
`HSM_KEY_SIZE_ECC_BP_R1_320` = 320,
`HSM_KEY_SIZE_ECC_BP_R1_384` = 384,
`HSM_KEY_SIZE_ECC_BP_R1_512` = 512,
`HSM_KEY_SIZE_ECC_NIST_224` = 224,
`HSM_KEY_SIZE_ECC_NIST_256` = 256,
`HSM_KEY_SIZE_ECC_NIST_384` = 384,
`HSM_KEY_SIZE_ECC_NIST_521` = 521,
`HSM_KEY_SIZE_ECC_BP_T1_224` = 224,
`HSM_KEY_SIZE_ECC_BP_T1_256` = 256,
`HSM_KEY_SIZE_ECC_BP_T1_320` = 320,
`HSM_KEY_SIZE_ECC_BP_T1_384` = 384 }
- enum `hsm_permitted_algo_t` {
`PERMITTED_ALGO_SHA224` = `ALGO_HASH_SHA224`,
`PERMITTED_ALGO_SHA256` = `ALGO_HASH_SHA256`,
`PERMITTED_ALGO_SHA384` = `ALGO_HASH_SHA384`,
`PERMITTED_ALGO_SHA512` = `ALGO_HASH_SHA512`,
`PERMITTED_ALGO_SM3` = `ALGO_HASH_SM3`,
`PERMITTED_ALGO_HMAC_SHA256` = `ALGO_HMAC_SHA256`,
`PERMITTED_ALGO_HMAC_SHA384` = `ALGO_HMAC_SHA384`,
`PERMITTED_ALGO_CMAC` = `ALGO_CMAC`,

```

PERMITTED_ALGO_CTR = ALGO_CIPHER_CTR,
PERMITTED_ALGO_CFB = ALGO_CIPHER_CFB,
PERMITTED_ALGO_OFB = ALGO_CIPHER_OFB,
PERMITTED_ALGO_ECB_NO_PADDING = ALGO_CIPHER_ECB_NO_PAD,
PERMITTED_ALGO_CBC_NO_PADDING = ALGO_CIPHER_CBC_NO_PAD,
PERMITTED_ALGO_CCM = ALGO_CCM,
PERMITTED_ALGO_GCM = ALGO_GCM,
PERMITTED_ALGO_RSA_PKCS1_V15_SHA224 = ALGO_RSA_PKCS1_V15_SHA224,
PERMITTED_ALGO_RSA_PKCS1_V15_SHA256 = ALGO_RSA_PKCS1_V15_SHA256,
PERMITTED_ALGO_RSA_PKCS1_V15_SHA384 = ALGO_RSA_PKCS1_V15_SHA384,
PERMITTED_ALGO_RSA_PKCS1_V15_SHA512 = ALGO_RSA_PKCS1_V15_SHA512,
PERMITTED_ALGO_RSA_PKCS1_PSS_MGF1_SHA224 = ALGO_RSA_PKCS1_PSS_MGF1_SHA224,
PERMITTED_ALGO_RSA_PKCS1_PSS_MGF1_SHA256 = ALGO_RSA_PKCS1_PSS_MGF1_SHA256,
PERMITTED_ALGO_RSA_PKCS1_PSS_MGF1_SHA384 = ALGO_RSA_PKCS1_PSS_MGF1_SHA384,
PERMITTED_ALGO_RSA_PKCS1_PSS_MGF1_SHA512 = ALGO_RSA_PKCS1_PSS_MGF1_SHA512,
PERMITTED_ALGO_ECDSA_SHA224 = ALGO_ECDSA_SHA224,
PERMITTED_ALGO_ECDSA_SHA256 = ALGO_ECDSA_SHA256,
PERMITTED_ALGO_ECDSA_SHA384 = ALGO_ECDSA_SHA384,
PERMITTED_ALGO_ECDSA_SHA512 = ALGO_ECDSA_SHA512,
PERMITTED_ALGO_HMAC_KDF_SHA256 = ALGO_HMAC_KDF_SHA256,
PERMITTED_ALGO_ALL_CIPHER = ALGO_CIPHER_ALL,
PERMITTED_ALGO_ALL_AEAD = ALGO_ALL_AEAD,
PERMITTED_ALGO_OTH_KEY_CBC = ALGO_CIPHER_KEY_CBC }
• enum hsm\_key\_lifecycle\_t {
    HSM_KEY_LIFECYCLE_OPEN = 0x1,
    HSM_KEY_LIFECYCLE_CLOSED = 0x2,
    HSM_KEY_LIFECYCLE_CLOSED_LOCKED = 0x4 }

```

Functions

- [hsm_err_t hsm_delete_key](#) ([hsm_hdl_t](#) key_management_hdl, [op_delete_key_args_t](#) *args)
- [hsm_err_t hsm_get_key_attr](#) ([hsm_hdl_t](#) key_management_hdl, [op_get_key_attr_args_t](#) *args)
- [hsm_err_t hsm_import_key](#) ([hsm_hdl_t](#) key_management_hdl, [op_import_key_args_t](#) *args)
- [hsm_err_t hsm_generate_key_ext](#) ([hsm_hdl_t](#) key_management_hdl, [op_generate_key_ext_args_t](#) *args)
- [hsm_err_t hsm_generate_key](#) ([hsm_hdl_t](#) key_management_hdl, [op_generate_key_args_t](#) *args)
- [hsm_err_t hsm_open_key_management_service](#) ([hsm_hdl_t](#) key_store_hdl, [open_svc_key_management_args_t](#) *args, [hsm_hdl_t](#) *key_management_hdl)
- [hsm_err_t hsm_close_key_management_service](#) ([hsm_hdl_t](#) key_management_hdl)

5.2.1 Detailed Description

5.2.2 Data Structure Documentation

5.2.2.1 [struct op_delete_key_args_t](#) Structure detailing the delete key operation member arguments

Data Fields

uint32_t	key_identifier	identifier of the key to be used for the operation.
hsm_op_delete_key_flags_t	flags	bitmap specifying the operation properties.

5.2.2.2 [struct op_get_key_attr_args_t](#) Structure describing the get key attribute operation arguments

Data Fields

uint32_t	key_identifier	identifier of the key to be used for the operation.
hsm_key_type_t	key_type	indicates which type of key must be generated.
hsm_bit_key_sz_t	bit_key_sz	
hsm_key_lifetime_t	key_lifetime	
hsm_key_usage_t	key_usage	
hsm_permitted_algo_t	permitted_algo	
hsm_key_lifecycle_t	lifecycle	

5.2.2.3 struct op_import_key_args_t Structure detailing the import key operation member arguments

Data Fields

uint32_t	key_identifier	Identifier of the KEK used to encrypt the key to be imported (Ignored if KEK is not used as set as part of "flags" field).
uint8_t *	input_lsb_addr	Address in the requester space where: <ul style="list-style-type: none"> • EdgeLock 2GO TLV can be found. • Ignore this field if not E2GO_TLV.
uint32_t	input_size	Size in bytes of: <ul style="list-style-type: none"> • EdgeLock 2GO TLV can be found. • Ignore this field if not E2GO_TLV.
hsm_op_import_key_flags_t	flags	bitmap specifying the operation properties.

5.2.2.4 struct kek_enc_key_hdr_t Structure describing the encryption key header

Data Fields

uint8_t	iv[IV_LENGTH]	
uint8_t *	key	
uint32_t	tag	

5.2.2.5 struct op_generate_key_ext_args_t Structure detailing the key generate operation member arguments

Data Fields

uint32_t *	key_identifier	pointer to the identifier of the key to be used for the operation In case of create operation the new key identifier will be stored in this location
uint16_t	out_size	length in bytes of the generated key It must be 0 in case of symmetric keys
hsm_op_key_gen_flags_t	flags	bitmap specifying the operation properties
hsm_key_type_t	key_type	indicates which type of key must be generated

Data Fields

hsm_key_group_t	key_group	Key group of the generated key. It must be a value in the range 0-1023. Keys belonging to the same group can be cached in the HSM local memory through the hsm_manage_key_group API
hsm_key_info_t	key_info	bitmap specifying the properties of the key
uint8_t *	out_key	pointer to the output area where the generated public key must be written.
uint8_t	min_mac_len	min mac length in bits to be set for this key, value 0 indicates use default (see op_mac_one_go_args_t for more details). Only accepted for keys that can be used for mac operations, must not be larger than maximum mac size that can be performed with the key. When in FIPS approved mode values < 32 bits are not allowed.
uint8_t	reserved[3]	It must be 0.

5.2.2.6 **struct op_generate_key_args_t** Structure describing the generate key operation member arguments

Data Fields

uint32_t *	key_identifier	pointer to the identifier of the key to be used for the operation. In case of create operation the new key identifier will be stored in this location.
uint16_t	out_size	length in bytes of the generated key. It must be 0 in case of symmetric keys.
hsm_op_key_gen_flags_t	flags	bitmap specifying the operation properties.
hsm_key_type_t	key_type	indicates which type of key must be generated.
hsm_key_group_t	key_group	Key group of the generated key. It must be a value in the range 0-1023. Keys belonging to the same group can be cached in the HSM local memory through the hsm_manage_key_group API.
uint8_t *	out_key	pointer to the output area where the generated public key must be written.
hsm_bit_key_sz_t	bit_key_sz	
hsm_key_lifecycle_t	key_lifecycle	defines the device lifecycle in which the key is usable. If it is set to 0, current device lifecycle is used.
hsm_key_lifetime_t	key_lifetime	
hsm_key_usage_t	key_usage	
hsm_permitted_algo_t	permitted_algo	

5.2.2.7 **struct open_svc_key_management_args_t** Structure detailing the key management open service member arguments

Data Fields

hsm_hdl_t	key_management_hdl	handle identifying the key management service flow
hsm_svc_key_management_flags_t	flags	bitmap specifying the services properties.

5.2.3 Macro Definition Documentation

5.2.3.1 HSM_OP_DEL_KEY_FLAGS_STRICT_OPERATION `#define HSM_OP_DEL_KEY_FLAGS_STRICT_OPERATION ((hsm_op_import_key_flags_t) (1u << 7))`

Bitmap detailing the delete key operation properties. Bit 0-6: Reserved. Bit 7: Strict: Request completed - New key written to NVM with updated MC.

5.2.3.2 HSM_KEY_USAGE_EXPORT `#define HSM_KEY_USAGE_EXPORT ((hsm_key_usage_t) (1u << 0))`

Bit indicating the permission to export the key

5.2.3.3 HSM_KEY_USAGE_ENCRYPT `#define HSM_KEY_USAGE_ENCRYPT ((hsm_key_usage_t) (1u << 8))`

Bit indicating the permission to encrypt a message with the key

5.2.3.4 HSM_KEY_USAGE_DECRYPT `#define HSM_KEY_USAGE_DECRYPT ((hsm_key_usage_t) (1u << 9))`

Bit indicating the permission to decrypt a message with the key

5.2.3.5 HSM_KEY_USAGE_SIGN_MSG `#define HSM_KEY_USAGE_SIGN_MSG ((hsm_key_usage_t) (1u << 10))`

Bit indicating the permission to sign a message with the key

5.2.3.6 HSM_KEY_USAGE_VERIFY_MSG `#define HSM_KEY_USAGE_VERIFY_MSG ((hsm_key_usage_t) (1u << 11))`

Bit indicating the permission to verify a message signature with the key

5.2.3.7 HSM_KEY_USAGE_SIGN_HASH `#define HSM_KEY_USAGE_SIGN_HASH ((hsm_key_usage_t) (1u << 12))`

Bit indicating the permission to sign a hashed message with the key

5.2.3.8 HSM_KEY_USAGE_VERIFY_HASH `#define HSM_KEY_USAGE_VERIFY_HASH ((hsm_key_usage_t) (1u << 13))`

Bit indicating the permission to verify a hashed message signature with the key

5.2.3.9 HSM_KEY_USAGE_DERIVE `#define HSM_KEY_USAGE_DERIVE ((hsm_key_usage_t) (1u << 14))`

Bit indicating the permission to derive other keys from this key

5.2.3.10 HSM_KEY_INFO_PERSISTENT `#define HSM_KEY_INFO_PERSISTENT ((hsm_key_info_t) (0u << 1))`

Bit indicating persistent keys which are stored in the external NVM. The entire key group is written in the NVM at the next STRICT operation.

5.2.3.11 HSM_KEY_INFO_PERMANENT `#define HSM_KEY_INFO_PERMANENT ((hsm_key_info_t) (1u << 0))`

Bit indicating the key is permanent. When set, the key is permanent (write locked). Once created, it will not be possible to update or delete the key anymore. Transient keys will be anyway deleted after a PoR or when the corresponding key store service flow is closed. This bit can never be reset.

5.2.3.12 HSM_KEY_INFO_TRANSIENT `#define HSM_KEY_INFO_TRANSIENT ((hsm_key_info_t) (1u << 1))`

Bit indicating the key is transient. Transient keys are deleted when the corresponding key store service flow is closed or after a PoR. Transient keys cannot be in the same key group than persistent keys.

5.2.3.13 HSM_KEY_INFO_MASTER `#define HSM_KEY_INFO_MASTER ((hsm_key_info_t) (1u << 2))`

Bit indicating the key is master key. When set, the key is considered as a master key. Only master keys can be used as input of key derivation functions (i.e butterfly key expansion).

5.2.3.14 HSM_KEY_INFO_KEK `#define HSM_KEY_INFO_KEK ((hsm_key_info_t) (1u << 3))`

Bit indicating the key is key encryption key. When set, the key is considered as a key encryption key. KEK keys can only be used to wrap and import other keys into the key store, all other operation are not allowed. Only keys imported in the key store through the hsm_manage_key API can get this attribute.

5.2.3.15 FLAG `#define FLAG 0`

structure defining

5.2.3.16 HSM_OP_KEY_GENERATION_FLAGS_STRICT_OPERATION `#define HSM_OP_KEY_GENERATION_FLAGS_STRICT_OPERATION ((hsm_op_key_gen_flags_t) (1u << 7))`

The request is completed only when the new key has been written in the NVM. This applicable for persistent key only.

5.2.4 Typedef Documentation

5.2.4.1 hsm_op_delete_key_flags_t `typedef uint8_t hsm_op_delete_key_flags_t`

Bitmap describing the delete key operation properties

5.2.4.2 hsm_op_import_key_flags_t typedef uint8_t [hsm_op_import_key_flags_t](#)

Bitmap specifying the import key operation supported properties Bit 0: Defines input configuration Bit 1-6: Reserved Bit 7: Strict

5.2.4.3 hsm_key_usage_t typedef uint32_t [hsm_key_usage_t](#)

Bitmap indicating the cryptographic operations that key can execute

5.2.4.4 hsm_key_group_t typedef uint16_t [hsm_key_group_t](#)

Bit field indicating the key group

5.2.4.5 hsm_key_info_t typedef uint16_t [hsm_key_info_t](#)

Bit field indicating the key information

5.2.4.6 hsm_op_key_gen_flags_t typedef uint8_t [hsm_op_key_gen_flags_t](#)

Reserverd Bits 0 - 6.

Bitmap specifying the key generate operation supported properties.

5.2.4.7 hsm_svc_key_management_flags_t typedef uint8_t [hsm_svc_key_management_flags_t](#)

Bitmap specifying the key management service supported properties

5.2.5 Enumeration Type Documentation**5.2.5.1 hsm_storage_loc_t** enum [hsm_storage_loc_t](#)

Enum Indicating the key location indicator.

5.2.5.2 hsm_storage_persist_lvl_t enum [hsm_storage_persist_lvl_t](#)

Enum Indicating the key persistent level indicator.

5.2.5.3 hsm_key_lifetime_t enum [hsm_key_lifetime_t](#)

Enum Indicating the key lifetime.

5.2.5.4 hsm_pubkey_type_t enum [hsm_pubkey_type_t](#)

Enum Indicating the public key type.

5.2.5.5 hsm_key_type_t enum [hsm_key_type_t](#)

Enum Indicating the key type.

5.2.5.6 hsm_bit_key_sz_t enum [hsm_bit_key_sz_t](#)

Enum Indicating the key security size in bits.

5.2.5.7 hsm_permitted_algo_t enum [hsm_permitted_algo_t](#)

Enum describing the permitted algorithm

5.2.5.8 hsm_key_lifecycle_t enum [hsm_key_lifecycle_t](#)

Enum detailing Permitted key lifecycle

5.2.6 Function Documentation**5.2.6.1 hsm_delete_key()** [hsm_err_t](#) [hsm_delete_key](#) (
 [hsm_hdl_t](#) *key_management_hdl*,
 [op_delete_key_args_t](#) * *args*)

This command is designed to perform the following operations:

- delete an existing key

Parameters

<i>key_management_hdl</i>	handle identifying the key management service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.2.6.2 hsm_get_key_attr() [hsm_err_t](#) [hsm_get_key_attr](#) (
 [hsm_hdl_t](#) *key_management_hdl*,
 [op_get_key_attr_args_t](#) * *args*)

This command is designed to perform the following operations:

- get attributes of an existing key

Parameters

<i>key_management_hdl</i>	handle identifying the key management service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

```
5.2.6.3 hsm_import_key() hsm_err_t hsm_import_key (
    hsm_hdl_t key_management_hdl,
    op_import_key_args_t * args )
```

This API will be used to Import the key

Parameters

<i>key_management_hdl</i>	handle identifying the key management service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

```
5.2.6.4 hsm_generate_key_ext() hsm_err_t hsm_generate_key_ext (
    hsm_hdl_t key_management_hdl,
    op_generate_key_ext_args_t * args )
```

Generate a key or a key pair with extended settings. Basic operation is identical to `hsm_generate_key`, but accepts additional settings. Currently the `min_mac_len` is the only additional setting accepted.

Parameters

<i>key_management_hdl</i>	handle identifying the key management service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.2.6.5 hsm_generate_key() `hsm_err_t hsm_generate_key (`
`hsm_hdl_t key_management_hdl,`
`op_generate_key_args_t * args)`

Generate a key or a key pair. Only the confidential keys (symmetric and private keys) are stored in the internal key store, while the non-confidential keys (public key) are exported.

The generated key can be stored using a new or existing key identifier with the restriction that an existing key can be replaced only by a key of the same type.

Parameters

<i>key_management_hdl</i>	handle identifying the key management service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.2.6.6 hsm_open_key_management_service() `hsm_err_t hsm_open_key_management_service (`
`hsm_hdl_t key_store_hdl,`
`open_svc_key_management_args_t * args,`
`hsm_hdl_t * key_management_hdl)`

Open a key management service flow

User must open this service flow in order to perform operation on the key store keys (generate, update, delete)

Parameters

<i>key_store_hdl</i>	handle identifying the key store service flow.
<i>args</i>	pointer to the structure containing the function arguments.
<i>key_management_hdl</i>	pointer to where the key management service flow handle must be written.

Returns

error_code error code.

5.2.6.7 hsm_close_key_management_service() `hsm_err_t hsm_close_key_management_service (`
`hsm_hdl_t key_management_hdl)`

Terminate a previously opened key management service flow

Parameters

<i>key_management_hdl</i>	handle identifying the key management service flow.
---------------------------	---

Returns

error code

5.3 Cipherring

Data Structures

- struct [op_auth_enc_args_t](#)
- struct [open_svc_cipher_args_t](#)
- struct [op_cipher_one_go_args_t](#)

Macros

- #define [HSM_AUTH_ENC_FLAGS_DECRYPT](#) ((hsm_op_auth_enc_flags_t)(0u << 0))
- #define [HSM_AUTH_ENC_FLAGS_ENCRYPT](#) ((hsm_op_auth_enc_flags_t)(1u << 0))
- #define [HSM_AUTH_ENC_FLAGS_GENERATE_FULL_IV](#) ((hsm_op_auth_enc_flags_t)(1u << 1))
- #define [HSM_AUTH_ENC_FLAGS_GENERATE_COUNTER_IV](#) ((hsm_op_auth_enc_flags_t)(1u << 2))
- #define [HSM_CIPHER_ONE_GO_FLAGS_DECRYPT](#) ((hsm_op_cipher_one_go_flags_t)(0u << 0))
- #define [HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT](#) ((hsm_op_cipher_one_go_flags_t)(1u << 0))

Typedefs

- typedef uint8_t [hsm_op_auth_enc_flags_t](#)
- typedef uint8_t [hsm_svc_cipher_flags_t](#)
- typedef uint8_t [hsm_op_cipher_one_go_flags_t](#)

Enumerations

- enum [hsm_op_auth_enc_algo_t](#) { [HSM_AEAD_ALGO_CCM](#) = ALGO_CCM }
- enum [hsm_op_cipher_one_go_algo_t](#) {
 [HSM_CIPHER_ONE_GO_ALGO_CTR](#) = ALGO_CIPHER_CTR,
 [HSM_CIPHER_ONE_GO_ALGO_CFB](#) = ALGO_CIPHER_CFB,
 [HSM_CIPHER_ONE_GO_ALGO_OFB](#) = ALGO_CIPHER_OFB,
 [HSM_CIPHER_ONE_GO_ALGO_ECB](#) = ALGO_CIPHER_ECB_NO_PAD,
 [HSM_CIPHER_ONE_GO_ALGO_CBC](#) = ALGO_CIPHER_CBC_NO_PAD }

Functions

- [hsm_err_t hsm_do_cipher](#) ([hsm_hdl_t](#) cipher_hdl, [op_cipher_one_go_args_t](#) *cipher_one_go)
- [hsm_err_t hsm_auth_enc](#) ([hsm_hdl_t](#) cipher_hdl, [op_auth_enc_args_t](#) *args)
- [hsm_err_t hsm_open_cipher_service](#) ([hsm_hdl_t](#) key_store_hdl, [open_svc_cipher_args_t](#) *args, [hsm_hdl_t](#) *cipher_hdl)
- [hsm_err_t hsm_cipher_one_go](#) ([hsm_hdl_t](#) cipher_hdl, [op_cipher_one_go_args_t](#) *args)
- [hsm_err_t hsm_close_cipher_service](#) ([hsm_hdl_t](#) cipher_hdl)

5.3.1 Detailed Description

5.3.2 Data Structure Documentation

5.3.2.1 struct [op_auth_enc_args_t](#) Structure describing the authenticated encryption operation arguments

Data Fields

uint32_t	key_identifier	< identifier of the key to be used for the operation pointer to the user supplied part of initialization vector or nonce,
uint8_t *	iv	length in bytes of the fixed part of the initialization vector for
uint16_t	iv_size	pointer to the additional authentication data
uint8_t *	aad	length in bytes of the additional authentication data
uint16_t	aad_size	algorithm to be used for the operation
hsm_op_auth_enc_algo_t	ae_algo	bitmap specifying the operation attributes
hsm_op_auth_enc_flags_t	flags	pointer to the input area plaintext for encryption
uint8_t *	input	pointer to the output area Ciphertext + Tag (16 bytes)
uint8_t *	output	length in bytes of the input
uint32_t	input_size	length in bytes of the output
uint32_t	output_size	

5.3.2.2 struct open_svc_cipher_args_t Structure describing the open cipher service members

Data Fields

hsm_hdl_t	cipher_hdl	handle identifying the cipher service flow
hsm_svc_cipher_flags_t	flags	bitmap specifying the services properties
uint8_t	reserved[3]	

5.3.2.3 struct op_cipher_one_go_args_t Structure describing the cipher one go operation arguments

Data Fields

uint32_t	key_identifier	< identifier of the key to be used for the operation pointer to the initialization vector (nonce in case of AES CCM)
uint8_t *	iv	length in bytes of the initialization vector.
uint16_t	iv_size	bitmap specifying the services properties.
hsm_svc_cipher_flags_t	svc_flags	bitmap specifying the operation attributes
hsm_op_cipher_one_go_flags_t	flags	algorithm to be used for the operation
hsm_op_cipher_one_go_algo_t	cipher_algo	pointer to the input area:
uint8_t *	input	pointer to the output area:
uint8_t *	output	length in bytes of the input.
uint32_t	input_size	length in bytes of the output
uint32_t	output_size	

5.3.3 Macro Definition Documentation

5.3.3.1 HSM_AUTH_ENC_FLAGS_DECRYPT `#define HSM_AUTH_ENC_FLAGS_DECRYPT ((hsm_op_auth_enc_flags_t) (0u << 0))`

Bit indicating the decryption operation

5.3.3.2 HSM_AUTH_ENC_FLAGS_ENCRYPT `#define HSM_AUTH_ENC_FLAGS_ENCRYPT ((hsm_op_auth_enc_flags_t) (1u << 0))`

Bit indicating the encryption operation

5.3.3.3 HSM_AUTH_ENC_FLAGS_GENERATE_FULL_IV `#define HSM_AUTH_ENC_FLAGS_GENERATE_FULL_IV ((hsm_op_auth_enc_flags_t) (1u << 1))`

Bit indicating the Full IV is internally generated (only relevant for encryption)

5.3.3.4 HSM_AUTH_ENC_FLAGS_GENERATE_COUNTER_IV `#define HSM_AUTH_ENC_FLAGS_GENERATE_COUNTER_IV ((hsm_op_auth_enc_flags_t) (1u << 2))`

Bit indicating 4 bytes supplied other bytes internally generated (only relevant for encryption)

5.3.3.5 HSM_CIPHER_ONE_GO_FLAGS_DECRYPT `#define HSM_CIPHER_ONE_GO_FLAGS_DECRYPT ((hsm_op_cipher_one_go_flags_t) (0u << 0))`

Bit indicating the decrypt operation

5.3.3.6 HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT `#define HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT ((hsm_op_cipher_one_go_flags_t) (0u << 0))`

Bit indicating the encrypt operation

5.3.4 Typedef Documentation

5.3.4.1 hsm_op_auth_enc_flags_t `typedef uint8_t hsm_op_auth_enc_flags_t`

Bit field indicating the authenticated encryption operations

5.3.4.2 hsm_svc_cipher_flags_t `typedef uint8_t hsm_svc_cipher_flags_t`

Bit field describing the open cipher service requested operation

5.3.4.3 hsm_op_cipher_one_go_flags_t `typedef uint8_t hsm_op_cipher_one_go_flags_t`

Bit field indicating the requested operations

5.3.5 Enumeration Type Documentation

5.3.5.1 hsm_op_auth_enc_algo_t `enum hsm_op_auth_enc_algo_t`

Bit field indicating the supported algorithm

Enumerator

HSM_AEAD_ALGO_CCM	CCM (AES CCM)
-------------------	---------------

5.3.5.2 hsm_op_cipher_one_go_algo_t enum hsm_op_cipher_one_go_algo_t

Enum describing the cipher one go operation algorithm

Enumerator

HSM_CIPHER_ONE_GO_ALGO_CTR	CTR (AES supported). CFB (AES supported).
HSM_CIPHER_ONE_GO_ALGO_CFB	OFB (AES supported).
HSM_CIPHER_ONE_GO_ALGO_OFB	ECB no padding (AES, SM4 supported).
HSM_CIPHER_ONE_GO_ALGO_ECB	CBC no padding (AES, SM4 supported).

5.3.6 Function Documentation

5.3.6.1 hsm_do_cipher() hsm_err_t hsm_do_cipher (hsm_hdl_t cipher_hdl, op_cipher_one_go_args_t * cipher_one_go)

Secondary API to perform ciphering operation

This API does the following:

1. Open an Cipher Service Flow
 2. Perform ciphering operation
 3. Terminate a previously opened cipher service flow
- User can call this function only after having opened a cipher service flow.

Parameters

<i>cipher_hdl</i>	handle identifying the cipher service flow.
<i>cipher_one_go</i>	pointer to the structure containing the function arguments.

Returns

error code

```

5.3.6.2 hsm_auth_enc() hsm_err_t hsm_auth_enc (
    hsm_hdl_t cipher_hdl,
    op_auth_enc_args_t * args )

```

Perform authenticated encryption operation

User can call this function only after having opened a cipher service flow

For decryption operations, the full IV is supplied by the caller via the iv and iv_size parameters. HSM_AUTH_ENC_FLAGS_GENERATE_FULL_IV and HSM_AUTH_ENC_FLAGS_GENERATE_COUNTER_IV flags are ignored. For encryption operations, either HSM_AUTH_ENC_FLAGS_GENERATE_FULL_IV or HSM_AUTH_ENC_FLAGS_GENERATE_COUNTER_IV must be set when calling this function:

- When HSM_AUTH_ENC_FLAGS_GENERATE_FULL_IV is set, the full IV is internally generated, iv and iv_size must be set to 0
- When HSM_AUTH_ENC_FLAGS_GENERATE_COUNTER_IV is set, the user supplies a 4 byte fixed part of the IV. The other IV bytes are internally generated

Parameters

<i>cipher_hdl</i>	handle identifying the cipher service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

```

5.3.6.3 hsm_open_cipher_service() hsm_err_t hsm_open_cipher_service (
    hsm_hdl_t key_store_hdl,
    open_svc_cipher_args_t * args,
    hsm_hdl_t * cipher_hdl )

```

- Open a cipher service flow.
- User can call this function only after having opened a key-store service flow.
- User must open this service in order to perform cipher operation.

Parameters

<i>key_store_hdl</i>	handle identifying the key store service flow.
<i>args</i>	pointer to the structure containing the function arguments.
<i>cipher_hdl</i>	pointer to where the cipher service flow handle must be written.

Returns

error code

5.3.6.4 hsm_cipher_one_go() `hsm_err_t hsm_cipher_one_go (`
`hsm_hdl_t cipher_hdl,`
`op_cipher_one_go_args_t * args)`

Perform cipherring operation

User can call this function only after having opened a cipher service flow

Parameters

<i><code>cipher_hdl</code></i>	handle identifying the cipher service flow.
<i><code>args</code></i>	pointer to the structure containing the function arguments.

Returns

error code

5.3.6.5 hsm_close_cipher_service() `hsm_err_t hsm_close_cipher_service (`
`hsm_hdl_t cipher_hdl)`

Terminate a previously opened cipher service flow

Parameters

<i><code>cipher_hdl</code></i>	pointer to handle identifying the cipher service flow to be closed.
--------------------------------	---

Returns

error code

5.4 Signature generation

Data Structures

- struct [open_svc_sign_gen_args_t](#)
- struct [op_generate_sign_args_t](#)
- struct [op_prepare_sign_args_t](#)

Macros

- #define [HSM_OP_GENERATE_SIGN_FLAGS_INPUT_DIGEST](#) ((hsm_op_generate_sign_flags_t)(0u << 0))
- #define [HSM_OP_GENERATE_SIGN_FLAGS_INPUT_MESSAGE](#) ((hsm_op_generate_sign_flags_t)(1u << 0))
- #define [HSM_OP_PREPARE_SIGN_INPUT_DIGEST](#) ((hsm_op_prepare_signature_flags_t)(0u << 0))
Bit indicating input digest.
- #define [HSM_OP_PREPARE_SIGN_INPUT_MESSAGE](#) ((hsm_op_prepare_signature_flags_t)(1u << 0))
Bit indicating input message.
- #define [HSM_OP_PREPARE_SIGN_COMPRESSED_POINT](#) ((hsm_op_prepare_signature_flags_t)(1u << 1))
Bit indicating compressed point.

Typedefs

- typedef uint8_t [hsm_op_generate_sign_flags_t](#)
- typedef uint8_t [hsm_op_prepare_signature_flags_t](#)

Enumerations

- enum [hsm_signature_scheme_id_t](#) {
[HSM_SIGNATURE_SCHEME_RSA_PKCS1_V15_SHA224](#) = 0x06000208,
[HSM_SIGNATURE_SCHEME_RSA_PKCS1_V15_SHA256](#) = 0x06000209,
[HSM_SIGNATURE_SCHEME_RSA_PKCS1_V15_SHA384](#) = 0x0600020A,
[HSM_SIGNATURE_SCHEME_RSA_PKCS1_V15_SHA512](#) = 0x0600020B,
[HSM_SIGNATURE_SCHEME_RSA_PKCS1_V15_ANY_HASH](#) = 0x060002FF,
[HSM_SIGNATURE_SCHEME_RSA_PKCS1_PSS_MGF1_SHA224](#) = 0x06000308,
[HSM_SIGNATURE_SCHEME_RSA_PKCS1_PSS_MGF1_SHA256](#) = 0x06000309,
[HSM_SIGNATURE_SCHEME_RSA_PKCS1_PSS_MGF1_SHA384](#) = 0x0600030A,
[HSM_SIGNATURE_SCHEME_RSA_PKCS1_PSS_MGF1_SHA512](#) = 0x0600030B,
[HSM_SIGNATURE_SCHEME_RSA_PKCS1_PSS_MGF1_ANY_HASH](#) = 0x060003FF,
[HSM_SIGNATURE_SCHEME_ECDSA_ANY](#) = 0x06000600,
[HSM_SIGNATURE_SCHEME_ECDSA_SHA224](#) = 0x06000608,
[HSM_SIGNATURE_SCHEME_ECDSA_SHA256](#) = 0x06000609,
[HSM_SIGNATURE_SCHEME_ECDSA_SHA384](#) = 0x0600060A,
[HSM_SIGNATURE_SCHEME_ECDSA_SHA512](#) = 0x0600060B }

Functions

- [hsm_err_t hsm_do_sign](#) ([hsm_hdl_t](#) key_store_hdl, [op_generate_sign_args_t](#) *args)
- [hsm_err_t hsm_open_signature_generation_service](#) ([hsm_hdl_t](#) key_store_hdl, [open_svc_sign_gen_args_t](#) *args, [hsm_hdl_t](#) *signature_gen_hdl)
- [hsm_err_t hsm_close_signature_generation_service](#) ([hsm_hdl_t](#) signature_gen_hdl)
- [hsm_err_t hsm_generate_signature](#) ([hsm_hdl_t](#) signature_gen_hdl, [op_generate_sign_args_t](#) *args)
- [hsm_err_t hsm_prepare_signature](#) ([hsm_hdl_t](#) signature_gen_hdl, [op_prepare_sign_args_t](#) *args)

5.4.1 Detailed Description

5.4.2 Data Structure Documentation

5.4.2.1 struct open_svc_sign_gen_args_t Structure to represent the generate sign open service arguments

Data Fields

hsm_hdl_t	signature_gen_hdl	
---------------------------	-------------------	--

5.4.2.2 struct op_generate_sign_args_t Structure to represent the generate sign operation arguments

Data Fields

uint32_t	key_identifier	< identifier of the key to be used for the operation pointer to the input (message or message digest) to be signed
uint8_t *	message	pointer to the output area where the signature must be stored.
uint8_t *	signature	length in bytes of the output. After signature generation operation,
uint16_t	signature_size	length in bytes of the input
uint32_t	message_size	identifier of the digital signature scheme to be used
hsm_signature_scheme_id_t	scheme_id	Salt length in bytes.
uint16_t	salt_len	expected signature buffer size for output, returned by FW in case
uint16_t	exp_signature_size	bitmap specifying the operation attributes
hsm_op_generate_sign_flags_t	flags	

5.4.2.3 struct op_prepare_sign_args_t Structure detailing the prepare signature operation member arguments

Data Fields

hsm_signature_scheme_id_t	scheme_id	identifier of the digital signature scheme to be used for the operation.
hsm_op_prepare_signature_flags_t	flags	bitmap specifying the operation attributes

5.4.3 Macro Definition Documentation

5.4.3.1 HSM_OP_GENERATE_SIGN_FLAGS_INPUT_DIGEST `#define HSM_OP_GENERATE_SIGN_FLAGS_INPUT_DIGEST ((hsm_op_generate_sign_flags_t) (0u << 0))`

Bit field indicating the input is the message digest

5.4.3.2 HSM_OP_GENERATE_SIGN_FLAGS_INPUT_MESSAGE `#define HSM_OP_GENERATE_SIGN_FLAGS_INPUT_MESSAGE ((hsm_op_generate_sign_flags_t) (1u << 0))`

Bit field indicating the input is the actual message

5.4.4 Typedef Documentation

5.4.4.1 hsm_op_generate_sign_flags_t `typedef uint8_t hsm_op_generate_sign_flags_t`

Bit field indicating the requested operation

5.4.4.2 hsm_op_prepare_signature_flags_t `typedef uint8_t hsm_op_prepare_signature_flags_t`

Bitmap specifying the prepare signature operation supported attributes

5.4.5 Enumeration Type Documentation

5.4.5.1 hsm_signature_scheme_id_t `enum hsm_signature_scheme_id_t`

Bit field indicating the PSA compliant requested operations: Bit 2 to 7: Reserved.

5.4.6 Function Documentation

5.4.6.1 hsm_do_sign() `hsm_err_t hsm_do_sign (hsm_hdl_t key_store_hdl, op_generate_sign_args_t * args)`

Secondary API to generate signature on the given message.
This API does the following:

1. Open a service flow for signature generation.
2. Based on the flag to identify the type of message: Digest or actual message, generate the signature using the key corresponding to the key id.
3. Post performing the operation, terminate the previously opened signature-generation service flow.
User can call this function only after having opened a key-store.

Parameters

<i>key_store_hdl</i>	handle identifying the current key-store.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.4.6.2 hsm_open_signature_generation_service() `hsm_err_t` hsm_open_signature_generation_service

```
(
    hsm_hdl_t key_store_hdl,
    open_svc_sign_gen_args_t * args,
    hsm_hdl_t * signature_gen_hdl )
```

Open a signature generation service flow

User can call this function only after having opened a key store service flow.

User must open this service in order to perform signature generation operations.

Parameters

<i>key_store_hdl</i>	handle identifying the key store service flow.
<i>args</i>	pointer to the structure containing the function arguments.
<i>signature_gen_hdl</i>	pointer to where the signature generation service flow handle must be written.

Returns

error code

5.4.6.3 hsm_close_signature_generation_service() `hsm_err_t` hsm_close_signature_generation_↵

```
service (
    hsm_hdl_t signature_gen_hdl )
```

Terminate a previously opened signature generation service flow

Parameters

<i>signature_gen_hdl</i>	handle identifying the signature generation service flow to be closed.
--------------------------	--

Returns

error code

5.4.6.4 hsm_generate_signature() `hsm_err_t hsm_generate_signature (`
`hsm_hdl_t signature_gen_hdl,`
`op_generate_sign_args_t * args)`

Generate a digital signature according to the signature scheme User can call this function only after having opened a signature generation service flow.

The signature $S=(r,s)$ is stored in the format $r||s||Ry$ where:

- Ry is an additional byte containing the lsb of y. Ry has to be considered valid only if the HSM_OP_GENERATE_SIGN_FLAGS_COMPRESSED_POINT is set.

In case of HSM_SIGNATURE_SCHEME_DSA_SM2_FP_256_SM3, message of `op_generate_sign_args_t` should be (as specified in GB/T 32918):

- equal to $Z||M$ in case of HSM_OP_GENERATE_SIGN_FLAGS_INPUT_MESSAGE
- equal to $SM3(Z||M)$ in case of HSM_OP_GENERATE_SIGN_FLAGS_INPUT_DIGEST

Parameters

<i>signature_gen_hdl</i>	handle identifying the signature generation service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.4.6.5 hsm_prepare_signature() `hsm_err_t hsm_prepare_signature (`
`hsm_hdl_t signature_gen_hdl,`
`op_prepare_sign_args_t * args)`

Prepare the creation of a signature by pre-calculating the operations having not dependencies on the input message.

The pre-calculated value will be stored internally and used once call `hsm_generate_signature`. Up to 20 pre-calculated values can be stored, additional preparation operations will have no effects.

User can call this function only after having opened a signature generation service flow.

The signature $S=(r,s)$ is stored in the format $r||s||Ry$ where:

- Ry is an additional byte containing the lsb of y, Ry has to be considered valid only if the HSM_OP_PREPARE_SIGN_FLAGS_COMPRESSED_POINT is set.

Parameters

<i>signature_gen_hdl</i>	handle identifying the signature generation service flow
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.5 Signature verification

Data Structures

- struct [open_svc_sign_ver_args_t](#)
- struct [op_verify_sign_args_t](#)

Macros

- #define [HSM_OP_VERIFY_SIGN_FLAGS_INPUT_DIGEST](#) (([hsm_op_verify_sign_flags_t](#))(0u << 0))
- #define [HSM_OP_VERIFY_SIGN_FLAGS_INPUT_MESSAGE](#) (([hsm_op_verify_sign_flags_t](#))(1u << 0))
- #define [HSM_OP_VERIFY_SIGN_FLAGS_COMPRESSED_POINT](#) (([hsm_op_verify_sign_flags_t](#))(1u << 1))
- #define [HSM_OP_VERIFY_SIGN_FLAGS_KEY_INTERNAL](#) (([hsm_op_verify_sign_flags_t](#))(1u << 2))
- #define [HSM_VERIFICATION_STATUS_SUCCESS](#) (([hsm_verification_status_t](#))(0x5A3CC3A5u))
- #define [HSM_VERIFICATION_STATUS_FAILURE](#) (([hsm_verification_status_t](#))(0x2B4DD4B2u))

Typedefs

- typedef uint32_t [hsm_verification_status_t](#)
- typedef uint8_t [hsm_op_verify_sign_flags_t](#)

Functions

- [hsm_err_t hsm_verify_sign](#) ([hsm_hdl_t](#) session_hdl, [op_verify_sign_args_t](#) *args, [hsm_verification_status_t](#) *verification_status)
- [hsm_err_t hsm_open_signature_verification_service](#) ([hsm_hdl_t](#) session_hdl, [open_svc_sign_ver_args_t](#) *args, [hsm_hdl_t](#) *signature_ver_hdl)
- [hsm_err_t hsm_close_signature_verification_service](#) ([hsm_hdl_t](#) signature_ver_hdl)
- [hsm_err_t hsm_verify_signature](#) ([hsm_hdl_t](#) signature_ver_hdl, [op_verify_sign_args_t](#) *args, [hsm_verification_status_t](#) *status)

5.5.1 Detailed Description

5.5.2 Data Structure Documentation

5.5.2.1 struct open_svc_sign_ver_args_t Structure to represent verify sign open service arguments

Data Fields

hsm_hdl_t	sig_ver_hdl	
---------------------------	-------------	--

5.5.2.2 struct op_verify_sign_args_t Structure to represent verify signature operation arguments

Data Fields

uint8_t *	key	< pointer to the public key to be used for the verification. pointer to the input (message or message digest)
-----------	-----	--

Data Fields

uint8_t *	message	pointer to the input signature. The signature S=(r,s) is expected
uint8_t *	signature	length in bytes of the input key
uint16_t	key_size	length in bytes of the output - it must contain one additional
uint16_t	signature_size	length in bytes of the input message
uint32_t	message_size	
hsm_verification_status_t	verification_status	identifier of the digital signature scheme to be used
hsm_signature_scheme_id_t	scheme_id	salt length in bytes
uint16_t	salt_len	
hsm_bit_key_sz_t	key_sz	
hsm_pubkey_type_t	pkey_type	bitmap specifying the operation attributes
hsm_op_verify_sign_flags_t	flags	

5.5.3 Macro Definition Documentation

5.5.3.1 HSM_OP_VERIFY_SIGN_FLAGS_INPUT_DIGEST #define HSM_OP_VERIFY_SIGN_FLAGS_INPUT_DIGEST ((hsm_op_verify_sign_flags_t) (0u << 0))

Verify signature bit indicating input is message digest

5.5.3.2 HSM_OP_VERIFY_SIGN_FLAGS_INPUT_MESSAGE #define HSM_OP_VERIFY_SIGN_FLAGS_INPUT_MESSAGE ((hsm_op_verify_sign_flags_t) (1u << 0))

Verify signature bit indicating input is actual message

5.5.3.3 HSM_OP_VERIFY_SIGN_FLAGS_COMPRESSED_POINT #define HSM_OP_VERIFY_SIGN_FLAGS_COMPRESSED_POINT ((hsm_op_verify_sign_flags_t) (1u << 1))

Verify signature bit indicating input based on signature format

5.5.3.4 HSM_OP_VERIFY_SIGN_FLAGS_KEY_INTERNAL #define HSM_OP_VERIFY_SIGN_FLAGS_KEY_INTERNAL ((hsm_op_verify_sign_flags_t) (1u << 2))

Verify signature bit indicating input is key argument

5.5.3.5 HSM_VERIFICATION_STATUS_SUCCESS #define HSM_VERIFICATION_STATUS_SUCCESS ((hsm_verification_status_t) A3CC3A5u)

Verify signature response success status

5.5.3.6 HSM_VERIFICATION_STATUS_FAILURE `#define HSM_VERIFICATION_STATUS_FAILURE ((hsm_verification_status_t) B4DD4B2u)`

Verify signature response failure status

5.5.4 Typedef Documentation

5.5.4.1 hsm_verification_status_t `typedef uint32_t hsm_verification_status_t`

Bit indicating the response verification status

5.5.4.2 hsm_op_verify_sign_flags_t `typedef uint8_t hsm_op_verify_sign_flags_t`

Bit indicating the requested operations

5.5.5 Function Documentation

5.5.5.1 hsm_verify_sign() `hsm_err_t hsm_verify_sign (hsm_hdl_t session_hdl, op_verify_sign_args_t * args, hsm_verification_status_t * verification_status)`

Secondary API to verify a message signature.

This API does the following:

1. Open a flow for verification of the signature.
2. Based on the flag to identify the type of message: Digest or actual message, verification of the signature is done using the public key.
3. Post performing the operation, terminate the previously opened signature-verification service flow.
User can call this function only after having opened a session.

Parameters

<i>session_hdl</i>	handle identifying the current key-store.
<i>args</i>	pointer to the structure containing the function arguments.
<i>verification_status</i>	pointer for storing the verification status.

Returns

error code

5.5.5.2 hsm_open_signature_verification_service() `hsm_err_t hsm_open_signature_verification_↔`
`service (`
`hsm_hdl_t session_hdl,`
`open_svc_sign_ver_args_t * args,`
`hsm_hdl_t * signature_ver_hdl)`

User must open this service in order to perform signature verification operations. User can call this function only after having opened a session.

Parameters

<i>session_hdl</i>	handle identifying the current session.
<i>args</i>	pointer to the structure containing the function arguments.
<i>signature_ver_hdl</i>	pointer to where the signature verification service flow handle must be written.

Returns

error code

5.5.5.3 hsm_close_signature_verification_service() `hsm_err_t hsm_close_signature_verification_↔`
`service (`
`hsm_hdl_t signature_ver_hdl)`

Terminate a previously opened signature verification service flow

Parameters

<i>signature_ver_hdl</i>	handle identifying the signature verification service flow to be closed.
--------------------------	--

Returns

error code

5.5.5.4 hsm_verify_signature() `hsm_err_t hsm_verify_signature (`
`hsm_hdl_t signature_ver_hdl,`
`op_verify_sign_args_t * args,`
`hsm_verification_status_t * status)`

Verify a digital signature according to the signature scheme User can call this function only after having opened a signature verification service flow.

The signature S=(r,s) is expected to be in format r||s||Ry where:

- Ry is an additional byte containing the lsb of y. Ry will be considered as valid only, if the HSM_OP_VERIFY_Y_SIGN_FLAGS_COMPRESSED_POINT is set.

Only not-compressed keys (x,y) can be used by this command. Compressed keys can be decompressed by using the dedicated API.

In case of HSM_SIGNATURE_SCHEME_DSA_SM2_FP_256_SM3, message of [op_verify_sign_args_t](#) should be (as specified in GB/T 32918):

- equal to $Z||M$ in case of HSM_OP_VERIFY_SIGN_FLAGS_INPUT_MESSAGE
- equal to $SM3(Z||M)$ in case of HSM_OP_VERIFY_SIGN_FLAGS_INPUT_DIGEST

Parameters

<i>signature_ver_hdl</i>	handle identifying the signature verification service flow.
<i>args</i>	pointer to the structure containing the function arguments.
<i>status</i>	pointer to where the verification status must be stored if the verification succeed the value HSM_VERIFICATION_STATUS_SUCCESS is returned.

Returns

error code

5.6 Random number generation

Data Structures

- struct [op_get_random_args_t](#)

Functions

- [hsm_err_t hsm_do_rng](#) ([hsm_hdl_t](#) session_hdl, [op_get_random_args_t](#) *args)
- [hsm_err_t hsm_get_random](#) ([hsm_hdl_t](#) rng_hdl, [op_get_random_args_t](#) *args)

5.6.1 Detailed Description

5.6.2 Data Structure Documentation

5.6.2.1 struct op_get_random_args_t Structure detailing the get random number operation member arguments

Data Fields

uint8_t *	output	pointer to the output area where the random number must be written
uint32_t	random_size	length in bytes of the random number to be provided.

5.6.3 Function Documentation

5.6.3.1 hsm_do_rng() [hsm_err_t](#) hsm_do_rng (
 [hsm_hdl_t](#) session_hdl,
 [op_get_random_args_t](#) * args)

Secondary API to fetch the Random Number

This API does the following: Get a freshly generated random number

Parameters

<i>session_hdl</i>	handle identifying the current session.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.6.3.2 hsm_get_random() `hsm_err_t hsm_get_random (`
`hsm_hdl_t rng_hdl,`
`op_get_random_args_t * args)`

Get a freshly generated random number

User can call this function only after having opened a rng service flow

Parameters

<i>rng_hdl</i>	handle identifying the rng service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.7 Hashing

Data Structures

- struct [op_hash_one_go_args_t](#)

Macros

- #define [HSM_HASH_FLAG_ALLOWED](#)

Enumerations

- enum [hsm_hash_algo_t](#) {
HSM_HASH_ALGO_SHA_224 = 0x02000008,
HSM_HASH_ALGO_SHA_256 = 0x02000009,
HSM_HASH_ALGO_SHA_384 = 0x0200000A,
HSM_HASH_ALGO_SHA_512 = 0x0200000B }
- enum [hsm_hash_svc_flags_t](#) {
HSM_HASH_FLAG_ONE_SHOT = 0x1,
HSM_HASH_FLAG_INIT = 0x2,
HSM_HASH_FLAG_UPDATE = 0x4,
HSM_HASH_FLAG_FINAL = 0x8,
HSM_HASH_FLAG_GET_CONTEXT = 0x80 }

Functions

- [hsm_err_t hsm_do_hash](#) ([hsm_hdl_t](#) session_hdl, [op_hash_one_go_args_t](#) *args)
- [hsm_err_t hsm_hash_one_go](#) ([hsm_hdl_t](#) hash_hdl, [op_hash_one_go_args_t](#) *args)

5.7.1 Detailed Description

5.7.2 Data Structure Documentation

5.7.2.1 struct [op_hash_one_go_args_t](#) Structure describing the hash one go operation arguments

Data Fields

uint8_t *	msb	< pointer to the MSB of address in the requester space where buffers pointer to the context.
uint8_t *	ctx	pointer to the input data to be hashed
uint8_t *	input	pointer to the output area where the resulting digest must be written
uint8_t *	output	length in bytes of the input
uint32_t	input_size	length in bytes of the output
uint32_t	output_size	hash algorithm to be used for the operation
hsm_hash_algo_t	algo	flags identifying the operation init() update(), final() or one shot
hsm_hash_svc_flags_t	svc_flags	size of context buffer in bytes, ignored in case of one shot
uint16_t	ctx_size	expected output digest buffer size, returned by FW in case the
uint32_t	exp_output_size	expected context size to allocate in bytes, if flag Get context
uint16_t	context_size	

5.7.3 Macro Definition Documentation

5.7.3.1 HSM_HASH_FLAG_ALLOWED `#define HSM_HASH_FLAG_ALLOWED`

Value:

```
(HSM_HASH_FLAG_ONE_SHOT | HSM_HASH_FLAG_INIT \
| HSM_HASH_FLAG_UPDATE | HSM_HASH_FLAG_FINAL \
| HSM_HASH_FLAG_GET_CONTEXT)
```

Bitmap indicating the allowed hash service operations

5.7.4 Enumeration Type Documentation

5.7.4.1 hsm_hash_algo_t `enum hsm_hash_algo_t`

Bitmap indicating the supported hash algorithm

5.7.4.2 hsm_hash_svc_flags_t `enum hsm_hash_svc_flags_t`

Bit field indicating the hash service operations

5.7.5 Function Documentation

5.7.5.1 hsm_do_hash() `hsm_err_t hsm_do_hash (` `hsm_hdl_t session_hdl,` `op_hash_one_go_args_t * args)`

Secondary API to digest a message.

This API does the following: Perform hash

Parameters

<i>session_hdl</i>	handle identifying the current session.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

```
5.7.5.2 hsm_hash_one_go() hsm_err_t hsm_hash_one_go (
    hsm_hdl_t hash_hdl,
    op_hash_one_go_args_t * args )
```

Perform the hash operation on a given input

User can call this function only after having opened a hash service flow

Parameters

<i>hash_hdl</i>	handle identifying the hash service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.8 Data storage

Data Structures

- struct [open_svc_data_storage_args_t](#)
- struct [op_data_storage_args_t](#)

Macros

- #define [HSM_OP_DATA_STORAGE_FLAGS_STORE](#) (([hsm_op_data_storage_flags_t](#))(1u << 0))
Store data.
- #define [HSM_OP_DATA_STORAGE_FLAGS_RETRIEVE](#) (([hsm_op_data_storage_flags_t](#))(0u << 0))
Retrieve data.

Typedefs

- typedef uint8_t [hsm_svc_data_storage_flags_t](#)
- typedef uint8_t [hsm_op_data_storage_flags_t](#)

Functions

- [hsm_err_t hsm_data_ops](#) ([hsm_hdl_t](#) key_store_hdl, [op_data_storage_args_t](#) *args)
- [hsm_err_t hsm_open_data_storage_service](#) ([hsm_hdl_t](#) key_store_hdl, [open_svc_data_storage_args_t](#) *args, [hsm_hdl_t](#) *data_storage_hdl)
- [hsm_err_t hsm_data_storage](#) ([hsm_hdl_t](#) data_storage_hdl, [op_data_storage_args_t](#) *args)
- [hsm_err_t hsm_close_data_storage_service](#) ([hsm_hdl_t](#) data_storage_hdl)

5.8.1 Detailed Description

5.8.2 Data Structure Documentation

5.8.2.1 struct open_svc_data_storage_args_t Structure specifying the data storage open service member arguments

Data Fields

hsm_hdl_t	data_storage_handle	
hsm_svc_data_storage_flags_t	flags	bitmap specifying the services properties.
uint8_t	reserved[3]	

5.8.2.2 struct op_data_storage_args_t Structure detailing the data storage operation member arguments

Data Fields

uint8_t *	data	< pointer to the data. In case of store request, length in bytes of the data
uint32_t	data_size	id of the data

Data Fields

uint16_t	data_id	bitmap specifying the services properties.
hsm_svc_data_storage_flags_t	flags	flags bitmap specifying the operation attributes.
hsm_op_data_storage_flags_t	svc_flags	

5.8.3 Typedef Documentation

5.8.3.1 [hsm_svc_data_storage_flags_t](#) typedef uint8_t [hsm_svc_data_storage_flags_t](#)

Bitmap specifying the data storage open service supported properties

5.8.3.2 [hsm_op_data_storage_flags_t](#) typedef uint8_t [hsm_op_data_storage_flags_t](#)

Bitmap specifying the data storage operation supported attributes

5.8.4 Function Documentation

5.8.4.1 [hsm_data_ops\(\)](#) [hsm_err_t](#) [hsm_data_ops](#) (
[hsm_hdl_t](#) [key_store_hdl](#),
[op_data_storage_args_t](#) * [args](#))

Secondary API to store and restore data from the linux filesystem managed by EdgeLock Enclave Firmware.

This API does the following:

1. Open an data storage service Flow
2. Based on the flag for operation attribute: Store or Re-store,
 - Store the data
 - Re-store the data, from the non-volatile storage.
3. Post performing the operation, terminate the previously opened data-storage service flow.
 User can call this function only after having opened a key-store.

Parameters

key_store_hdl	handle identifying the current key-store.
args	pointer to the structure containing the function arguments.

Returns

error code

5.8.4.2 hsm_open_data_storage_service() `hsm_err_t hsm_open_data_storage_service (`
`hsm_hdl_t key_store_hdl,`
`open_svc_data_storage_args_t * args,`
`hsm_hdl_t * data_storage_hdl)`

Open a data storage service flow

User must open this service flow in order to store/retrieve generic data in/from the HSM.

Parameters

<i>key_store_hdl</i>	handle identifying the key store service flow.
<i>args</i>	pointer to the structure containing the function arguments.
<i>data_storage_hdl</i>	pointer to where the data storage service flow handle must be written.

Returns

error_code error code.

5.8.4.3 hsm_data_storage() `hsm_err_t hsm_data_storage (`
`hsm_hdl_t data_storage_hdl,`
`op_data_storage_args_t * args)`

Store or retrieve generic data identified by a data_id.

Parameters

<i>data_storage_hdl</i>	handle identifying the data storage service flow.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.8.4.4 hsm_close_data_storage_service() `hsm_err_t hsm_close_data_storage_service (`
`hsm_hdl_t data_storage_hdl)`

Terminate a previously opened data storage service flow

Parameters

<i>data_storage_hdl</i>	handle identifying the data storage service flow.
-------------------------	---

Returns

error code

5.9 Authenticated Encryption

Functions

- [hsm_err_t hsm_do_auth_enc](#) ([hsm_hdl_t](#) key_store_hdl, [op_auth_enc_args_t](#) *auth_enc_args)

5.9.1 Detailed Description

5.9.2 Function Documentation

5.9.2.1 hsm_do_auth_enc() [hsm_err_t](#) hsm_do_auth_enc (
 [hsm_hdl_t](#) key_store_hdl,
 [op_auth_enc_args_t](#) * auth_enc_args)

Secondary API to perform Authenticated Encryption
This API does the following:

1. Opens Cipher Service Flow
2. Perform Authenticated Encryption operation
3. Terminates the previously opened Cipher service flow
User can call this function only after having opened a key store service flow.

Parameters

<i>key_store_hdl</i>	handle identifying the key store service flow.
<i>auth_enc_args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.10 Mac

Data Structures

- struct [open_svc_mac_args_t](#)
- struct [op_mac_one_go_args_t](#)

Macros

- #define [HSM_OP_MAC_ONE_GO_FLAGS_MAC_VERIFICATION](#) (([hsm_op_mac_one_go_flags_t](#))(0u << 0))
- #define [HSM_OP_MAC_ONE_GO_FLAGS_MAC_GENERATION](#) (([hsm_op_mac_one_go_flags_t](#))(1u << 0))
- #define [HSM_MAC_VERIFICATION_STATUS_SUCCESS](#) (([hsm_mac_verification_status_t](#))(0x6C1AA1↔C6u))

Typedefs

- typedef uint8_t [hsm_op_mac_one_go_flags_t](#)
 - typedef uint32_t [hsm_mac_verification_status_t](#)
 - typedef [hsm_permitted_algo_t](#) [hsm_op_mac_one_go_algo_t](#)
- < Following three permitted algos are allowed:

Functions

- [hsm_err_t](#) [hsm_do_mac](#) ([hsm_hdl_t](#) key_store_hdl, [op_mac_one_go_args_t](#) *mac_one_go)
- [hsm_err_t](#) [hsm_open_mac_service](#) ([hsm_hdl_t](#) key_store_hdl, [open_svc_mac_args_t](#) *args, [hsm_hdl_t](#) *mac_hdl)
- [hsm_err_t](#) [hsm_mac_one_go](#) ([hsm_hdl_t](#) mac_hdl, [op_mac_one_go_args_t](#) *args, [hsm_mac_verification_status_t](#) *status)
- [hsm_err_t](#) [hsm_close_mac_service](#) ([hsm_hdl_t](#) mac_hdl)

5.10.1 Detailed Description

5.10.2 Data Structure Documentation

5.10.2.1 struct open_svc_mac_args_t Structure describing the mac open service member arguments

Data Fields

hsm_hdl_t	mac_serv_hdl	
---------------------------	--------------	--

5.10.2.2 struct op_mac_one_go_args_t Structure describing the mac one go operation member arguments

Data Fields

uint32_t	key_identifier	< identifier of the key to be used for the operation algorithm to be used for the operation
----------	----------------	---

Data Fields

hsm_op_mac_one_go_algo_t	algorithm	bitmap specifying the operation attributes
hsm_op_mac_one_go_flags_t	flags	pointer to the payload area
uint8_t *	payload	pointer to the tag area
uint8_t *	mac	length in bytes of the payload
uint32_t	payload_size	length of the tag.
uint16_t	mac_size	expected mac size for output, returned by FW in case the mac size
uint16_t	expected_mac_size	
hsm_mac_verification_status_t	verification_status	

5.10.3 Macro Definition Documentation

5.10.3.1 HSM_OP_MAC_ONE_GO_FLAGS_MAC_VERIFICATION `#define HSM_OP_MAC_ONE_GO_FLAGS_MAC_VERIFICATION ((hsm_op_mac_one_go_flags_t) (0u << 0))`

Bit indicating mac one go verify operation

5.10.3.2 HSM_OP_MAC_ONE_GO_FLAGS_MAC_GENERATION `#define HSM_OP_MAC_ONE_GO_FLAGS_MAC_GENERATION ((hsm_op_mac_one_go_flags_t) (1u << 0))`

Bit indicating mac one go generate operation

5.10.3.3 HSM_MAC_VERIFICATION_STATUS_SUCCESS `#define HSM_MAC_VERIFICATION_STATUS_SUCCESS ((hsm_mac_verification_status_t) (0x6C1AA1C6u))`

Bit indicating mac verification success status

5.10.4 Typedef Documentation

5.10.4.1 hsm_op_mac_one_go_flags_t `typedef uint8_t hsm_op_mac_one_go_flags_t`

Bitmap describing the mac one go operation

5.10.4.2 hsm_mac_verification_status_t `typedef uint32_t hsm_mac_verification_status_t`

Bitmap describing the mac verification status

5.10.4.3 hsm_op_mac_one_go_algo_t typedef `hsm_permitted_algo_t hsm_op_mac_one_go_algo_t`

< Following three permitted algos are allowed:

Bitmap describing the mac one go operation permitted algorithm

5.10.5 Function Documentation

5.10.5.1 hsm_do_mac() `hsm_err_t hsm_do_mac (`
`hsm_hdl_t key_store_hdl,`
`op_mac_one_go_args_t * mac_one_go)`

Secondary API to perform mac operation

This API does the following:

1. Open an MAC Service Flow
2. Perform mac operation
3. Terminate a previously opened mac service flow
 User can call this function only after having opened a key store service flow.

Parameters

<i>key_store_hdl</i>	handle identifying the key store service flow.
<i>mac_one_go</i>	pointer to the structure containing the function arguments.

Returns

error code

5.10.5.2 hsm_open_mac_service() `hsm_err_t hsm_open_mac_service (`
`hsm_hdl_t key_store_hdl,`
`open_svc_mac_args_t * args,`
`hsm_hdl_t * mac_hdl)`

Open a mac service flow

User can call this function only after having opened a key store service flow.

User must open this service in order to perform mac operation

Parameters

<i>key_store_hdl</i>	handle identifying the key store service flow.
<i>args</i>	pointer to the structure containing the function arguments.
<i>mac_hdl</i>	pointer to where the mac service flow handle must be written.

Returns

error code

5.10.5.3 hsm_mac_one_go() `hsm_err_t hsm_mac_one_go (`
`hsm_hdl_t mac_hdl,`
`op_mac_one_go_args_t * args,`
`hsm_mac_verification_status_t * status)`

Perform mac operation

User can call this function only after having opened a mac service flow

For CMAC algorithm, a key of type HSM_KEY_TYPE_AES_XXX must be used

For HMAC algorithm, a key of type HSM_KEY_TYPE_HMAC_XXX must be used

For mac verification operations, the verified mac length can be specified in:

- Bits by setting the HSM_OP_MAC_ONE_GO_FLAGS_MAC_LENGTH_IN_BITS flag,
- if this flag is clear then the mac_length is specified in bytes.

For mac generation operations:

- mac length must be set in bytes, and
- HSM_OP_MAC_ONE_GO_FLAGS_MAC_LENGTH_IN_BITS flag must be 0

Parameters

<i>mac_hdl</i>	handle identifying the mac service flow.
<i>args</i>	pointer to the structure containing the function arguments.
<i>status</i>	pointer for storing the verification status.

Returns

error code

5.10.5.4 hsm_close_mac_service() `hsm_err_t hsm_close_mac_service (`
`hsm_hdl_t mac_hdl)`

Terminate a previously opened mac service flow

Parameters

<i>mac_hdl</i>	pointer to handle identifying the mac service flow to be closed.
----------------	--

Returns

error code

5.11 Dump Firmware Log

Data Structures

- struct [op_debug_dump_args_t](#)

Functions

- [hsm_err_t dump_firmware_log](#) ([hsm_hdl_t](#) session_hdl)

5.11.1 Detailed Description

5.11.2 Data Structure Documentation

5.11.2.1 struct op_debug_dump_args_t Structure detailing the debug dump operation member arguments

Data Fields

bool	is_dump_pending	
uint32_t	dump_buf_len	
uint32_t	dump_buf[MAC_BUFF_LEN]	

5.11.3 Function Documentation

5.11.3.1 dump_firmware_log() [hsm_err_t](#) dump_firmware_log ([hsm_hdl_t](#) session_hdl)

This command is designed to dump the firmware logs

Parameters

<i>session_hdl</i>	handle identifying the session handle.
--------------------	--

Returns

error code

5.12 Dev attest

Data Structures

- struct [op_dev_attest_args_t](#)

Functions

- [hsm_err_t hsm_dev_attest](#) ([hsm_hdl_t](#) sess_hdl, [op_dev_attest_args_t](#) *args)

5.12.1 Detailed Description

5.12.2 Data Structure Documentation

5.12.2.1 struct op_dev_attest_args_t Structure describing the device attestation operation member arguments

Data Fields

uint16_t	soc_id	
uint16_t	soc_rev	
uint16_t	lmda_val	
uint8_t	ssm_state	
uint8_t	uid_sz	
uint8_t *	uid	
uint16_t	rom_patch_sha_sz	
uint16_t	sha_fw_sz	
uint8_t *	sha_rom_patch	
uint8_t *	sha_fw	
uint32_t	nounce	
uint32_t	rsp_nounce	
uint8_t	attest_result	
uint8_t	reserved	
uint16_t	sign_sz	
uint8_t *	signature	

5.12.3 Function Documentation

5.12.3.1 hsm_dev_attest() [hsm_err_t](#) hsm_dev_attest (
[hsm_hdl_t](#) sess_hdl,
[op_dev_attest_args_t](#) * args)

Perform device attestation operation
 User can call this function only after having opened the session.

Parameters

<i>sess_hdl</i>	handle identifying the active session.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.13 Dev Info

Data Structures

- struct [op_dev_getinfo_args_t](#)

Functions

- [hsm_err_t hsm_dev_getinfo](#) ([hsm_hdl_t](#) sess_hdl, [op_dev_getinfo_args_t](#) *args)

5.13.1 Detailed Description

5.13.2 Data Structure Documentation

5.13.2.1 struct op_dev_getinfo_args_t Structure detailing the device getinfo operation member arguments

Data Fields

uint16_t	soc_id	
uint16_t	soc_rev	
uint16_t	lmda_val	
uint8_t	ssm_state	
uint8_t	uid_sz	
uint8_t *	uid	
uint16_t	rom_patch_sha_sz	
uint16_t	sha_fw_sz	
uint8_t *	sha_rom_patch	
uint8_t *	sha_fw	
uint16_t	oem_srkh_sz	
uint8_t *	oem_srkh	
uint8_t	imem_state	
uint8_t	csal_state	
uint8_t	trng_state	

5.13.3 Function Documentation

5.13.3.1 hsm_dev_getinfo() [hsm_err_t](#) hsm_dev_getinfo ([hsm_hdl_t](#) sess_hdl, [op_dev_getinfo_args_t](#) * args)

Perform device attestation operation

User can call this function only after having opened the session.

Parameters

sess_hdl	handle identifying the active session.
args	pointer to the structure containing the function arguments.

Returns

error code

5.14 Generic Crypto: Asymmetric Crypto

Data Structures

- struct [op_gc_crypto_args_t](#)

Macros

- #define [HSM_OP_GC_ACRYPTO_FLAGS_INPUT_MESSAGE](#) ((hsm_op_gc_acrypto_flags_t)(1u << 0))
- #define [HSM_GC_ACRYPTO_VERIFICATION_SUCCESS](#) ((hsm_gc_acrypto_verification_status_t)(0x5↵A3CC3A5u))
- #define [HSM_GC_ACRYPTO_VERIFICATION_FAILURE](#) ((hsm_gc_acrypto_verification_status_t)(0x2B4↵DD4B2u))

Typedefs

- typedef uint8_t [hsm_op_gc_acrypto_flags_t](#)
- typedef uint32_t [hsm_gc_acrypto_verification_status_t](#)

Enumerations

- enum [hsm_op_gc_acrypto_algo_t](#) {
[HSM_GC_ACRYPTO_ALGO_ECDSA_SHA224](#) = ALGO_ECDSA_SHA224,
[HSM_GC_ACRYPTO_ALGO_ECDSA_SHA256](#) = ALGO_ECDSA_SHA256,
[HSM_GC_ACRYPTO_ALGO_ECDSA_SHA384](#) = ALGO_ECDSA_SHA384,
[HSM_GC_ACRYPTO_ALGO_ECDSA_SHA512](#) = ALGO_ECDSA_SHA512,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_V15_SHA224](#) = ALGO_RSA_PKCS1_V15_SHA224,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_V15_SHA256](#) = ALGO_RSA_PKCS1_V15_SHA256,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_V15_SHA384](#) = ALGO_RSA_PKCS1_V15_SHA384,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_V15_SHA512](#) = ALGO_RSA_PKCS1_V15_SHA512,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_PSS_MGF1_SHA224](#) = ALGO_RSA_PKCS1_PSS_MGF1 ↵
SHA224,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_PSS_MGF1_SHA256](#) = ALGO_RSA_PKCS1_PSS_MGF1 ↵
SHA256,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_PSS_MGF1_SHA384](#) = ALGO_RSA_PKCS1_PSS_MGF1 ↵
SHA384,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_PSS_MGF1_SHA512](#) = ALGO_RSA_PKCS1_PSS_MGF1 ↵
SHA512,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_V15_CRYPT](#) = ALGO_RSA_PKCS1_V15_CRYPT,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_OAEP_SHA1](#) = ALGO_RSA_PKCS1_OAEP_SHA1,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_OAEP_SHA224](#) = ALGO_RSA_PKCS1_OAEP_SHA224,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_OAEP_SHA256](#) = ALGO_RSA_PKCS1_OAEP_SHA256,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_OAEP_SHA384](#) = ALGO_RSA_PKCS1_OAEP_SHA384,
[HSM_GC_ACRYPTO_ALGO_RSA_PKCS1_OAEP_SHA512](#) = ALGO_RSA_PKCS1_OAEP_SHA512 }
- enum [hsm_gc_acrypto_op_mode_t](#) {
[HSM_GC_ACRYPTO_OP_MODE_ENCRYPT](#) = 0x01,
[HSM_GC_ACRYPTO_OP_MODE_DECRYPT](#) = 0x02,
[HSM_GC_ACRYPTO_OP_MODE_SIGN_GEN](#) = 0x03,
[HSM_GC_ACRYPTO_OP_MODE_SIGN_VER](#) = 0x04 }

Functions

- [hsm_err_t hsm_gc_acrypto](#) ([hsm_hdl_t](#) session_hdl, [op_gc_crypto_args_t](#) *args)

5.14.1 Detailed Description

5.14.2 Data Structure Documentation

5.14.2.1 **struct op_gc_acrypto_args_t** Structure describing the generic asymmetric crypto member arguments

Data Fields

<code>hsm_op_gc_acrypto_algo_t</code>	algorithm	< algorithm to use for the operation indicates the operation mode
<code>hsm_gc_acrypto_op_mode_t</code>	op_mode	indicates operation flags
<code>hsm_op_gc_acrypto_flags_t</code>	flags	key size in bits
<code>hsm_bit_key_sz_t</code>	bit_key_sz	pointer to the data buffer 1:
<code>uint8_t *</code>	data_buff1	pointer to the data buffer 2:
<code>uint8_t *</code>	data_buff2	size in bytes of data buffer 1
<code>uint32_t</code>	data_buff1_size	size in bytes of data buffer 2
<code>uint32_t</code>	data_buff2_size	pointer to the key modulus buffer
<code>uint8_t *</code>	key_buff1	pointer the key exponent, either private or public
<code>uint8_t *</code>	key_buff2	size in bytes of the key buffer 1
<code>uint16_t</code>	key_buff1_size	size in bytes of the key buffer 2
<code>uint16_t</code>	key_buff2_size	RSA label address.
<code>uint8_t *</code>	rsa_label	RSA label size in bytes.
<code>uint16_t</code>	rsa_label_size	RSA salt length in bytes.
<code>uint16_t</code>	rsa_salt_len	expected plaintext length in bytes, returned by FW in case of
<code>uint32_t</code>	exp_plaintext_len	signature verification status
<code>hsm_gc_acrypto_verification_status_t</code>	verification_status	

5.14.3 Macro Definition Documentation

5.14.3.1 HSM_OP_GC_ACRYPTO_FLAGS_INPUT_MESSAGE `#define HSM_OP_GC_ACRYPTO_FLAGS_INPUT_MESSAGE ((hsm_op_gc_acrypto_flags_t) (1u << 0))`

Bit indicating the generic asymmetric crypto input message operation

5.14.3.2 HSM_GC_ACRYPTO_VERIFICATION_SUCCESS `#define HSM_GC_ACRYPTO_VERIFICATION_SUCCESS ((hsm_gc_acrypto_verification_status_t) (0x5A3CC3A5u))`

Bit indicating the generic asymmetric crypto success verification status

5.14.3.3 HSM_GC_ACRYPTO_VERIFICATION_FAILURE `#define HSM_GC_ACRYPTO_VERIFICATION_FAILURE ((hsm_gc_acrypto_verification_status_t) (0x2B4DD4B2u))`

Bit indicating the generic asymmetric crypto failure verification status

5.14.4 Typedef Documentation

5.14.4.1 hsm_op_gc_acrypto_flags_t typedef uint8_t hsm_op_gc_acrypto_flags_t

Bitmap describing the generic asymmetric crypto supported operation

5.14.4.2 hsm_gc_acrypto_verification_status_t typedef uint32_t hsm_gc_acrypto_verification_status_t

Bitmap describing the generic asymmetric crypto verification status

5.14.5 Enumeration Type Documentation

5.14.5.1 hsm_op_gc_acrypto_algo_t enum hsm_op_gc_acrypto_algo_t

Enum detailing the generic asymmetric crypto supported algorithms

5.14.5.2 hsm_gc_acrypto_op_mode_t enum hsm_gc_acrypto_op_mode_t

Enum describing the generic asymmetric crypto supported operating modes

5.14.6 Function Documentation

5.14.6.1 hsm_gc_acrypto() hsm_err_t hsm_gc_acrypto (
 hsm_hdl_t session_hdl,
 op_gc_acrypto_args_t * args)

This command is designed to perform the following operations: -Asymmetric crypto -encryption/decryption - signature generation/verification

Parameters

session_hdl	handle identifying the current session.
args	pointer to the structure containing the function arguments.

Returns

error code

5.15 Generic Crypto Asymmetric Key Generate

Data Structures

- struct [op_gc_akey_gen_args_t](#)

Functions

- [hsm_err_t hsm_gc_akey_gen](#) ([hsm_hdl_t](#) session_hdl, [op_gc_akey_gen_args_t](#) *args)

5.15.1 Detailed Description

5.15.2 Data Structure Documentation

5.15.2.1 struct op_gc_akey_gen_args_t Structue detailing the generic crypto asymmetric key generate operation members

Data Fields

uint8_t *	modulus	pointer to the output buffer of key modulus
uint8_t *	priv_buff	pointer to the output buffer of key private exponent
uint8_t *	pub_buff	pointer to the input buffer containing key public exponent
uint16_t	modulus_size	size in bytes of the modulus buffer
uint16_t	priv_buff_size	size in bytes of the private exponent buffer
uint16_t	pub_buff_size	size in bytes of the public exponent buffer
hsm_key_type_t	key_type	indicates which type of keypair must be generated
hsm_bit_key_sz_t	bit_key_sz	size in bits of the keypair to be generated

5.15.3 Function Documentation

5.15.3.1 hsm_gc_akey_gen() [hsm_err_t](#) hsm_gc_akey_gen (
[hsm_hdl_t](#) session_hdl,
[op_gc_akey_gen_args_t](#) * args)

This command is designed to perform the following operations: -Generate asymmetric keys, without using FW keystore

Parameters

<i>session_hdl</i>	handle identifying the current session.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.16 Get Info

Data Structures

- struct [op_get_info_args_t](#)

Functions

- [hsm_err_t hsm_get_info](#) ([hsm_hdl_t](#) sess_hdl, [op_get_info_args_t](#) *args)

5.16.1 Detailed Description

5.16.2 Data Structure Documentation

5.16.2.1 struct op_get_info_args_t Structure describing the get info operation member arguments

Data Fields

uint32_t	user_sab_id	< Stores User identifier (32bits) Stores the chip unique identifier
uint8_t *	chip_unique_id	Size of the chip unique identifier in bytes.
uint16_t	chip_unq_id_sz	Stores the chip monotonic counter value (16bits)
uint16_t	chip_monotonic_counter	Stores the chip current life cycle bitfield (16bits)
uint16_t	chip_life_cycle	Stores the module version (32bits)
uint32_t	version	Stores the module extended version (32bits)
uint32_t	version_ext	Stores the FIPS mode bitfield (8bits). Bitmask definition: bit0 - FIPS mode of operation: <ul style="list-style-type: none"> • value 0 - part is running in FIPS non-approved mode. • value 1 - part is running in FIPS approved mode. bit1 - FIPS certified part: <ul style="list-style-type: none"> • value 0 - part is not FIPS certified. • value 1 - part is FIPS certified. bit2-7: reserved <ul style="list-style-type: none"> • value 0.
uint8_t	fips_mode	

5.16.3 Function Documentation

5.16.3.1 hsm_get_info() `hsm_err_t hsm_get_info (`
 `hsm_hdl_t sess_hdl,`
 `op_get_info_args_t * args)`

Perform device attestation operation

User can call this function only after having opened the session.

Parameters

<i>sess_hdl</i>	handle identifying the active session.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.17 Public key recovery

Public Key Recovery is now also known as Public Key Exportation, in PSA compliant APIs. The naming here has been kept unchanged, for backward compatibility and Non-PSA compliant APIs.

Data Structures

- struct [op_pub_key_recovery_args_t](#)

Functions

- [hsm_err_t hsm_pub_key_recovery](#) ([hsm_hdl_t](#) key_store_hdl, [op_pub_key_recovery_args_t](#) *args)

5.17.1 Detailed Description

Public Key Recovery is now also known as Public Key Exportation, in PSA compliant APIs. The naming here has been kept unchanged, for backward compatibility and Non-PSA compliant APIs.

5.17.2 Data Structure Documentation

5.17.2.1 struct op_pub_key_recovery_args_t Structure detailing the public key recovery operation member arguments

Data Fields

uint32_t	key_identifier	< pointer to the identifier of the key to be used for the operation pointer to the output area where the generated public key must be written
uint8_t *	out_key	length in bytes of the output key
uint16_t	out_key_size	

5.17.3 Function Documentation

5.17.3.1 hsm_pub_key_recovery() [hsm_err_t](#) hsm_pub_key_recovery (
[hsm_hdl_t](#) key_store_hdl,
[op_pub_key_recovery_args_t](#) * args)

Recover Public key from private key present in key store
 User can call this function only after having opened a key store.

Parameters

<i>key_store_hdl</i>	handle identifying the current key store.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.18 Key store

User must open a key store service flow in order to perform the following operations:

Data Structures

- struct `open_svc_key_store_args_t`

Macros

- #define `HSM_SVC_KEY_STORE_FLAGS_LOAD` `((hsm_svc_key_store_flags_t)(0u << 0))`
It must be specified to load a previously created key store.
- #define `HSM_SVC_KEY_STORE_FLAGS_CREATE` `((hsm_svc_key_store_flags_t)(1u << 0))`
- #define `HSM_SVC_KEY_STORE_FLAGS_SET_MAC_LEN` `((hsm_svc_key_store_flags_t)(1u << 3))`
- #define `HSM_SVC_KEY_STORE_FLAGS_STRICT_OPERATION` `((hsm_svc_key_store_flags_t)(1u << 7))`

Typedefs

- typedef uint8_t `hsm_svc_key_store_flags_t`

Functions

- `hsm_err_t hsm_open_key_store_service` (`hsm_hdl_t session_hdl`, `open_svc_key_store_args_t *args`, `hsm_hdl_t *key_store_hdl`)
- `hsm_err_t hsm_close_key_store_service` (`hsm_hdl_t key_store_hdl`)

5.18.1 Detailed Description

User must open a key store service flow in order to perform the following operations:

- create a new key store
- perform operations involving keys stored in the key store (ciphering, signature generation...)
- perform a key store reprovisioning using a signed message. A key store re-provisioning results in erasing all the key stores handled by the HSM.

To grant access to the key store, the caller is authenticated against the domain ID (DID) and Messaging Unit used at the keystore creation, additionally an authentication nonce can be provided.

5.18.2 Data Structure Documentation

5.18.2.1 struct open_svc_key_store_args_t Structure specifying the open key store service member arguments

Data Fields

hsm_hdl_t	key_store_hdl	< handle identifying the key store service flow user defined id identifying the key store.
uint32_t	key_store_identifier	user defined nonce used as authentication proof for accessing the
uint32_t	authentication_nonce	maximum number of updates authorized for the key store.
uint16_t	max_updates_number	bitmap specifying the services properties.
hsm_svc_key_store_flags_t	flags	pointer to signed_message to be sent only in case of
uint8_t *	signed_message	size of the signed_message to be sent only in case of
uint16_t	signed_msg_size	

5.18.3 Macro Definition Documentation

5.18.3.1 HSM_SVC_KEY_STORE_FLAGS_CREATE `#define HSM_SVC_KEY_STORE_FLAGS_CREATE ((hsm_svc_key_store_flags_t) 0)`

It must be specified to create a new key store. The key store will be stored in the NVM only if the STRICT OPERATION flag is set.

5.18.3.2 HSM_SVC_KEY_STORE_FLAGS_SET_MAC_LEN `#define HSM_SVC_KEY_STORE_FLAGS_SET_MAC_LEN ((hsm_svc_key_store_flags_t) (1u << 3))`

If set, minimum mac length specified in min_mac_length field will be stored in the key store when creating the key store.

Must only be set at key store creation.

5.18.3.3 HSM_SVC_KEY_STORE_FLAGS_STRICT_OPERATION `#define HSM_SVC_KEY_STORE_FLAGS_STRICT_OPERATION ((hsm_svc_key_store_flags_t) (1u << 7))`

The request is completed only when the new key store has been written in in the NVM. This applicable for CREATE operations only.

5.18.4 Typedef Documentation

5.18.4.1 hsm_svc_key_store_flags_t `typedef uint8_t hsm_svc_key_store_flags_t`

Bitmap specifying the open key store service supported attributes

5.18.5 Function Documentation

5.18.5.1 hsm_open_key_store_service() `hsm_err_t hsm_open_key_store_service (hsm_hdl_t session_hdl, open_svc_key_store_args_t * args, hsm_hdl_t * key_store_hdl)`

Open a service flow on the specified key store. Only one key store service can be opened on a given key store.

Parameters

<i>session_hdl</i>	pointer to the handle identifying the current session.
<i>args</i>	pointer to the structure containing the function arguments.
<i>key_store_hdl</i>	pointer to where the key store service flow handle must be written.

Returns

error_code error code.

5.18.5.2 hsm_close_key_store_service() `hsm_err_t hsm_close_key_store_service (hsm_hdl_t key_store_hdl)`

Close a previously opened key store service flow. The key store is deleted from the HSM local memory, any update not written in the NVM is lost

Parameters

<i>key_store_hdl</i>	handle identifying the key store service flow to be closed.
----------------------	---

Returns

error_code error code.

5.19 Life Cycle update

Data Structures

- struct [op_lc_update_msg_args_t](#)

Enumerations

- enum [hsm_lc_new_state_t](#) {
HSM_NXP_PROVISIONED_STATE = (1u << 0),
HSM_OEM_OPEN_STATE = (1u << 1),
HSM_OEM_CLOSE_STATE = (1u << 3),
HSM_OEM_FIELD_RET_STATE = (1u << 4),
HSM_NXP_FIELD_RET_STATE = (1u << 5),
HSM_OEM_LOCKED_STATE = (1u << 7) }

Functions

- [hsm_err_t hsm_lc_update](#) ([hsm_hdl_t](#) session_hdl, [op_lc_update_msg_args_t](#) *args)

5.19.1 Detailed Description

5.19.2 Data Structure Documentation

5.19.2.1 struct [op_lc_update_msg_args_t](#) Structure specifying the life cycle update message arguments

Data Fields

hsm_lc_new_state_t	new_lc_state	
------------------------------------	--------------	--

5.19.3 Enumeration Type Documentation

5.19.3.1 [hsm_lc_new_state_t](#) enum [hsm_lc_new_state_t](#)

Enum specifying the Life Cycle state

5.19.4 Function Documentation

5.19.4.1 [hsm_lc_update\(\)](#) [hsm_err_t](#) [hsm_lc_update](#) ([hsm_hdl_t](#) session_hdl, [op_lc_update_msg_args_t](#) * args)

This API will perform the Life Cycle update

Parameters

<i>session_hdl</i>	handle identifying the session handle.
<i>args</i>	pointer to the structure containing the function arguments.

Returns

error code

5.20 Error codes

Enumerations

```

• enum hsm_err_t {
    HSM_NO_ERROR = 0x0,
    HSM_INVALID_MESSAGE = 0x1,
    HSM_INVALID_ADDRESS = 0x2,
    HSM_UNKNOWN_ID = 0x3,
    HSM_INVALID_PARAM = 0x4,
    HSM_NVM_ERROR = 0x5,
    HSM_OUT_OF_MEMORY = 0x6,
    HSM_UNKNOWN_HANDLE = 0x7,
    HSM_UNKNOWN_KEY_STORE = 0x8,
    HSM_KEY_STORE_AUTH = 0x9,
    HSM_KEY_STORE_ERROR = 0xA,
    HSM_ID_CONFLICT = 0xB,
    HSM_RNG_NOT_STARTED = 0xC,
    HSM_CMD_NOT_SUPPORTED = 0xD,
    HSM_INVALID_LIFECYCLE = 0xE,
    HSM_KEY_STORE_CONFLICT = 0xF,
    HSM_KEY_STORE_COUNTER = 0x10,
    HSM_FEATURE_NOT_SUPPORTED = 0x11,
    HSM_SELF_TEST_FAILURE = 0x12,
    HSM_NOT_READY_RATING = 0x13,
    HSM_FEATURE_DISABLED = 0x14,
    HSM_KEY_GROUP_FULL = 0x19,
    HSM_CANNOT_RETRIEVE_KEY_GROUP = 0x1A,
    HSM_KEY_NOT_SUPPORTED = 0x1B,
    HSM_CANNOT_DELETE_PERMANENT_KEY = 0x1C,
    HSM_OUT_TOO_SMALL = 0x1D,
    HSM_CRC_CHECK_ERR = 0xB9,
    HSM_OEM_CLOSED_LC_SIGNED_MSG_VERIFICATION_FAIL = 0xF0,
    HSM_OEM_OPEN_LC_SIGNED_MSG_VERIFICATION_FAIL = 0xF0,
    HSM_FATAL_FAILURE = 0x29,
    HSM_SERVICES_DISABLED = 0xF4,
    HSM_UNKNOWN_WARNING = 0xFC,
    HSM_SIGNATURE_INVALID = 0xFD,
    HSM_UNKNOWN_ERROR = 0xFE,
    HSM_GENERAL_ERROR = 0xFF }

```

5.20.1 Detailed Description

5.20.2 Enumeration Type Documentation

5.20.2.1 hsm_err_t enum hsm_err_t

Error codes returned by HSM functions.

Enumerator

HSM_NO_ERROR	Success. The received message is invalid or unknown.
--------------	--

Enumerator

HSM_INVALID_MESSAGE	The provided address is invalid or doesn't respect the API requirements.
HSM_INVALID_ADDRESS	The provided identifier is not known.
HSM_UNKNOWN_ID	One of the parameter provided in the command is invalid.
HSM_INVALID_PARAM	NVM generic issue.
HSM_NVM_ERROR	There is not enough memory to handle the requested operation.
HSM_OUT_OF_MEMORY	Unknown session/service handle.
HSM_UNKNOWN_HANDLE	The key store identified by the provided "key store Id" doesn't exist and the "create" flag is not set.
HSM_UNKNOWN_KEY_STORE	Key store authentication fails.
HSM_KEY_STORE_AUTH	An error occurred in the key store internal processing.
HSM_KEY_STORE_ERROR	An element (key store, key. . .) with the provided ID already exists.
HSM_ID_CONFLICT	The internal RNG is not started.
HSM_RNG_NOT_STARTED	The functionality is not supported for the current session/service/key store configuration.
HSM_CMD_NOT_SUPPORTED	Invalid lifecycle for requested operation.
HSM_INVALID_LIFECYCLE	A key store with the same attributes already exists.
HSM_KEY_STORE_CONFLICT	The current key store reaches the max number of monotonic counter updates, updates are still allowed but monotonic counter will not be blown.
HSM_KEY_STORE_COUNTER	The requested feature is not supported by the firmware.
HSM_FEATURE_NOT_SUPPORTED	Self tests report an issue
HSM_SELF_TEST_FAILURE	The HSM is not ready to handle the current request
HSM_NOT_READY_RATING	The required service/operation is disabled
HSM_FEATURE_DISABLED	Not enough space to store the key in the key group
HSM_KEY_GROUP_FULL	Impossible to retrieve key group
HSM_CANNOT_RETRIEVE_KEY_GROUP	Key not supported
HSM_KEY_NOT_SUPPORTED	Trying to delete a permanent key
HSM_CANNOT_DELETE_PERMANENT_KEY	Output buffer size is too small
HSM_OUT_TOO_SMALL	Command CRC check error
HSM_CRC_CHECK_ERR	In OEM closed lifecycle, Signed message signature verification failure
HSM_OEM_CLOSED_LC_SIGNED_MSG_VERIFICATION_FAIL	Warning: In OEM open lifecycles, Signed message signature verification failure
HSM_OEM_OPEN_LC_SIGNED_MSG_VERIFICATION_FAIL	A fatal failure occurred, the HSM goes in unrecoverable error state not replying to further requests
HSM_FATAL_FAILURE	Message neither handled by ROM nor FW
HSM_SERVICES_DISABLED	Unknown warnings
HSM_UNKNOWN_WARNING	Failure in verification status of operations such as MAC verification, Signature verification.
HSM_SIGNATURE_INVALID	Unknown errors
HSM_UNKNOWN_ERROR	Error in case General Error is received

Index

- add_service
 - Session, [9](#)
- add_session
 - Session, [8](#)
- Authenticated Encryption, [48](#)
 - hsm_do_auth_enc, [48](#)
- Ciphering, [22](#)
 - HSM_AEAD_ALGO_CCM, [25](#)
 - hsm_auth_enc, [25](#)
 - HSM_AUTH_ENC_FLAGS_DECRYPT, [23](#)
 - HSM_AUTH_ENC_FLAGS_ENCRYPT, [24](#)
 - HSM_AUTH_ENC_FLAGS_GENERATE_COUNTER_IV, [24](#)
 - HSM_AUTH_ENC_FLAGS_GENERATE_FULL_IV, [24](#)
 - hsm_cipher_one_go, [26](#)
 - HSM_CIPHER_ONE_GO_ALGO_CFB, [25](#)
 - HSM_CIPHER_ONE_GO_ALGO_CTR, [25](#)
 - HSM_CIPHER_ONE_GO_ALGO_ECB, [25](#)
 - HSM_CIPHER_ONE_GO_ALGO_OFB, [25](#)
 - HSM_CIPHER_ONE_GO_FLAGS_DECRYPT, [24](#)
 - HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT, [24](#)
 - hsm_close_cipher_service, [27](#)
 - hsm_do_cipher, [25](#)
 - hsm_op_auth_enc_algo_t, [24](#)
 - hsm_op_auth_enc_flags_t, [24](#)
 - hsm_op_cipher_one_go_algo_t, [25](#)
 - hsm_op_cipher_one_go_flags_t, [24](#)
 - hsm_open_cipher_service, [26](#)
 - hsm_svc_cipher_flags_t, [24](#)
- Data storage, [44](#)
 - hsm_close_data_storage_service, [46](#)
 - hsm_data_ops, [45](#)
 - hsm_data_storage, [46](#)
 - hsm_op_data_storage_flags_t, [45](#)
 - hsm_open_data_storage_service, [46](#)
 - hsm_svc_data_storage_flags_t, [45](#)
- delete_service
 - Session, [8](#)
- delete_session
 - Session, [8](#)
- Dev attest, [55](#)
 - hsm_dev_attest, [55](#)
- Dev Info, [57](#)
 - hsm_dev_getinfo, [57](#)
- Dump Firmware Log, [54](#)
 - dump_firmware_log, [54](#)
- dump_firmware_log
 - Dump Firmware Log, [54](#)
- Error codes, [74](#)
 - HSM_CANNOT_DELETE_PERMANENT_KEY, [75](#)
 - HSM_CANNOT_RETRIEVE_KEY_GROUP, [75](#)
 - HSM_CMD_NOT_SUPPORTED, [75](#)
 - HSM_CRC_CHECK_ERR, [75](#)
 - hsm_err_t, [74](#)
 - HSM_FATAL_FAILURE, [75](#)
 - HSM_FEATURE_DISABLED, [75](#)
 - HSM_FEATURE_NOT_SUPPORTED, [75](#)
 - HSM_ID_CONFLICT, [75](#)
 - HSM_INVALID_ADDRESS, [75](#)
 - HSM_INVALID_LIFECYCLE, [75](#)
 - HSM_INVALID_MESSAGE, [75](#)
 - HSM_INVALID_PARAM, [75](#)
 - HSM_KEY_GROUP_FULL, [75](#)
 - HSM_KEY_NOT_SUPPORTED, [75](#)
 - HSM_KEY_STORE_AUTH, [75](#)
 - HSM_KEY_STORE_CONFLICT, [75](#)
 - HSM_KEY_STORE_COUNTER, [75](#)
 - HSM_KEY_STORE_ERROR, [75](#)
 - HSM_NO_ERROR, [74](#)
 - HSM_NOT_READY_RATING, [75](#)
 - HSM_NVM_ERROR, [75](#)
 - HSM_OEM_CLOSED_LC_SIGNED_MSG_VERIFICATION_FAIL, [75](#)
 - HSM_OEM_OPEN_LC_SIGNED_MSG_VERIFICATION_FAIL, [75](#)
 - HSM_OUT_OF_MEMORY, [75](#)
 - HSM_OUT_TOO_SMALL, [75](#)
 - HSM_RNG_NOT_STARTED, [75](#)
 - HSM_SELF_TEST_FAILURE, [75](#)
 - HSM_SERVICES_DISABLED, [75](#)
 - HSM_SIGNATURE_INVALID, [75](#)
 - HSM_UNKNOWN_ERROR, [75](#)
 - HSM_UNKNOWN_HANDLE, [75](#)
 - HSM_UNKNOWN_ID, [75](#)
 - HSM_UNKNOWN_KEY_STORE, [75](#)
 - HSM_UNKNOWN_WARNING, [75](#)
- FLAG
 - Key management, [16](#)
- Generic Crypto Asymmetric Key Generate, [63](#)
 - hsm_gc_akey_gen, [63](#)
- Generic Crypto: Asymmetric Crypto, [59](#)
 - hsm_gc_acrypto, [62](#)
 - hsm_gc_acrypto_op_mode_t, [62](#)
 - HSM_GC_ACRYPTO_VERIFICATION_FAILURE, [61](#)
 - hsm_gc_acrypto_verification_status_t, [62](#)
 - HSM_GC_ACRYPTO_VERIFICATION_SUCCESS, [61](#)
 - hsm_op_gc_acrypto_algo_t, [62](#)
 - HSM_OP_GC_ACRYPTO_FLAGS_INPUT_MESSAGE, [61](#)
 - hsm_op_gc_acrypto_flags_t, [61](#)
- Get Info, [65](#)
 - hsm_get_info, [65](#)
- Hashing, [41](#)

- hsm_do_hash, [42](#)
- hsm_hash_algo_t, [42](#)
- HSM_HASH_FLAG_ALLOWED, [42](#)
- hsm_hash_one_go, [42](#)
- hsm_hash_svc_flags_t, [42](#)
- HSM_AEAD_ALGO_CCM
 - Ciphering, [25](#)
- hsm_auth_enc
 - Ciphering, [25](#)
- HSM_AUTH_ENC_FLAGS_DECRYPT
 - Ciphering, [23](#)
- HSM_AUTH_ENC_FLAGS_ENCRYPT
 - Ciphering, [24](#)
- HSM_AUTH_ENC_FLAGS_GENERATE_COUNTER_IV
 - Ciphering, [24](#)
- HSM_AUTH_ENC_FLAGS_GENERATE_FULL_IV
 - Ciphering, [24](#)
- hsm_bit_key_sz_t
 - Key management, [18](#)
- HSM_CANNOT_DELETE_PERMANENT_KEY
 - Error codes, [75](#)
- HSM_CANNOT_RETRIEVE_KEY_GROUP
 - Error codes, [75](#)
- hsm_cipher_one_go
 - Ciphering, [26](#)
- HSM_CIPHER_ONE_GO_ALGO_CFB
 - Ciphering, [25](#)
- HSM_CIPHER_ONE_GO_ALGO_CTR
 - Ciphering, [25](#)
- HSM_CIPHER_ONE_GO_ALGO_ECB
 - Ciphering, [25](#)
- HSM_CIPHER_ONE_GO_ALGO_OFB
 - Ciphering, [25](#)
- HSM_CIPHER_ONE_GO_FLAGS_DECRYPT
 - Ciphering, [24](#)
- HSM_CIPHER_ONE_GO_FLAGS_ENCRYPT
 - Ciphering, [24](#)
- hsm_close_cipher_service
 - Ciphering, [27](#)
- hsm_close_data_storage_service
 - Data storage, [46](#)
- hsm_close_key_management_service
 - Key management, [20](#)
- hsm_close_key_store_service
 - Key store, [71](#)
- hsm_close_mac_service
 - Mac, [52](#)
- hsm_close_session
 - Session, [7](#)
- hsm_close_signature_generation_service
 - Signature generation, [31](#)
- hsm_close_signature_verification_service
 - Signature verification, [37](#)
- HSM_CMD_NOT_SUPPORTED
 - Error codes, [75](#)
- HSM_CRC_CHECK_ERR
 - Error codes, [75](#)
- hsm_data_ops
 - Data storage, [45](#)
- hsm_data_storage
 - Data storage, [46](#)
- hsm_delete_key
 - Key management, [18](#)
- hsm_dev_attest
 - Dev attest, [55](#)
- hsm_dev_getinfo
 - Dev Info, [57](#)
- hsm_do_auth_enc
 - Authenticated Encryption, [48](#)
- hsm_do_cipher
 - Ciphering, [25](#)
- hsm_do_hash
 - Hashing, [42](#)
- hsm_do_mac
 - Mac, [51](#)
- hsm_do_rng
 - Random number generation, [39](#)
- hsm_do_sign
 - Signature generation, [30](#)
- hsm_err_t
 - Error codes, [74](#)
- HSM_FATAL_FAILURE
 - Error codes, [75](#)
- HSM_FEATURE_DISABLED
 - Error codes, [75](#)
- HSM_FEATURE_NOT_SUPPORTED
 - Error codes, [75](#)
- hsm_gc_acrypto
 - Generic Crypto: Asymmetric Crypto, [62](#)
- hsm_gc_acrypto_op_mode_t
 - Generic Crypto: Asymmetric Crypto, [62](#)
- HSM_GC_ACRYPTO_VERIFICATION_FAILURE
 - Generic Crypto: Asymmetric Crypto, [61](#)
- hsm_gc_acrypto_verification_status_t
 - Generic Crypto: Asymmetric Crypto, [62](#)
- HSM_GC_ACRYPTO_VERIFICATION_SUCCESS
 - Generic Crypto: Asymmetric Crypto, [61](#)
- hsm_gc_akey_gen
 - Generic Crypto Asymmetric Key Generate, [63](#)
- hsm_generate_key
 - Key management, [19](#)
- hsm_generate_key_ext
 - Key management, [19](#)
- hsm_generate_signature
 - Signature generation, [31](#)
- hsm_get_info
 - Get Info, [65](#)
- hsm_get_key_attr
 - Key management, [18](#)
- hsm_get_random
 - Random number generation, [39](#)
- hsm_hash_algo_t
 - Hashing, [42](#)
- HSM_HASH_FLAG_ALLOWED
 - Hashing, [42](#)
- hsm_hash_one_go

- Hashing, [42](#)
- hsm_hash_svc_flags_t
 - Hashing, [42](#)
- hsm_hdl_t
 - Session, [6](#)
- HSM_ID_CONFLICT
 - Error codes, [75](#)
- hsm_import_key
 - Key management, [19](#)
- HSM_INVALID_ADDRESS
 - Error codes, [75](#)
- HSM_INVALID_LIFECYCLE
 - Error codes, [75](#)
- HSM_INVALID_MESSAGE
 - Error codes, [75](#)
- HSM_INVALID_PARAM
 - Error codes, [75](#)
- HSM_KEY_GROUP_FULL
 - Error codes, [75](#)
- hsm_key_group_t
 - Key management, [17](#)
- HSM_KEY_INFO_KEY
 - Key management, [16](#)
- HSM_KEY_INFO_MASTER
 - Key management, [16](#)
- HSM_KEY_INFO_PERMANENT
 - Key management, [16](#)
- HSM_KEY_INFO_PERSISTENT
 - Key management, [15](#)
- hsm_key_info_t
 - Key management, [17](#)
- HSM_KEY_INFO_TRANSIENT
 - Key management, [16](#)
- hsm_key_lifecycle_t
 - Key management, [18](#)
- hsm_key_lifetime_t
 - Key management, [17](#)
- HSM_KEY_NOT_SUPPORTED
 - Error codes, [75](#)
- HSM_KEY_STORE_AUTH
 - Error codes, [75](#)
- HSM_KEY_STORE_CONFLICT
 - Error codes, [75](#)
- HSM_KEY_STORE_COUNTER
 - Error codes, [75](#)
- HSM_KEY_STORE_ERROR
 - Error codes, [75](#)
- hsm_key_type_t
 - Key management, [17](#)
- HSM_KEY_USAGE_DECRYPT
 - Key management, [15](#)
- HSM_KEY_USAGE_DERIVE
 - Key management, [15](#)
- HSM_KEY_USAGE_ENCRYPT
 - Key management, [15](#)
- HSM_KEY_USAGE_EXPORT
 - Key management, [15](#)
- HSM_KEY_USAGE_SIGN_HASH
 - Key management, [15](#)
- HSM_KEY_USAGE_SIGN_MSG
 - Key management, [15](#)
- hsm_key_usage_t
 - Key management, [17](#)
- HSM_KEY_USAGE_VERIFY_HASH
 - Key management, [15](#)
- HSM_KEY_USAGE_VERIFY_MSG
 - Key management, [15](#)
- hsm_lc_new_state_t
 - Life Cycle update, [72](#)
- hsm_lc_update
 - Life Cycle update, [72](#)
- hsm_mac_one_go
 - Mac, [52](#)
- HSM_MAC_VERIFICATION_STATUS_SUCCESS
 - Mac, [50](#)
- hsm_mac_verification_status_t
 - Mac, [50](#)
- HSM_NO_ERROR
 - Error codes, [74](#)
- HSM_NOT_READY_RATING
 - Error codes, [75](#)
- HSM_NVM_ERROR
 - Error codes, [75](#)
- HSM_OEM_CLOSED_LC_SIGNED_MSG_VERIFICATION_FAIL
 - Error codes, [75](#)
- HSM_OEM_OPEN_LC_SIGNED_MSG_VERIFICATION_FAIL
 - Error codes, [75](#)
- hsm_op_auth_enc_algo_t
 - Ciphering, [24](#)
- hsm_op_auth_enc_flags_t
 - Ciphering, [24](#)
- hsm_op_cipher_one_go_algo_t
 - Ciphering, [25](#)
- hsm_op_cipher_one_go_flags_t
 - Ciphering, [24](#)
- hsm_op_data_storage_flags_t
 - Data storage, [45](#)
- HSM_OP_DEL_KEY_FLAGS_STRICT_OPERATION
 - Key management, [15](#)
- hsm_op_delete_key_flags_t
 - Key management, [16](#)
- hsm_op_gc_acrypto_algo_t
 - Generic Crypto: Asymmetric Crypto, [62](#)
- HSM_OP_GC_ACRYPTO_FLAGS_INPUT_MESSAGE
 - Generic Crypto: Asymmetric Crypto, [61](#)
- hsm_op_gc_acrypto_flags_t
 - Generic Crypto: Asymmetric Crypto, [61](#)
- HSM_OP_GENERATE_SIGN_FLAGS_INPUT_DIGEST
 - Signature generation, [29](#)
- HSM_OP_GENERATE_SIGN_FLAGS_INPUT_MESSAGE
 - Signature generation, [29](#)
- hsm_op_generate_sign_flags_t
 - Signature generation, [30](#)
- hsm_op_import_key_flags_t
 - Key management, [16](#)
- hsm_op_key_gen_flags_t

- Key management, 17
- HSM_OP_KEY_GENERATION_FLAGS_STRICT_OPERATION
 - Key management, 16
- hsm_op_mac_one_go_algo_t
 - Mac, 50
- HSM_OP_MAC_ONE_GO_FLAGS_MAC_GENERATION
 - Mac, 50
- HSM_OP_MAC_ONE_GO_FLAGS_MAC_VERIFICATION
 - Mac, 50
- hsm_op_mac_one_go_flags_t
 - Mac, 50
- hsm_op_prepare_signature_flags_t
 - Signature generation, 30
- HSM_OP_VERIFY_SIGN_FLAGS_COMPRESSED_POINT
 - Signature verification, 35
- HSM_OP_VERIFY_SIGN_FLAGS_INPUT_DIGEST
 - Signature verification, 35
- HSM_OP_VERIFY_SIGN_FLAGS_INPUT_MESSAGE
 - Signature verification, 35
- HSM_OP_VERIFY_SIGN_FLAGS_KEY_INTERNAL
 - Signature verification, 35
- hsm_op_verify_sign_flags_t
 - Signature verification, 36
- hsm_open_cipher_service
 - Ciphering, 26
- hsm_open_data_storage_service
 - Data storage, 46
- hsm_open_key_management_service
 - Key management, 20
- hsm_open_key_store_service
 - Key store, 70
- hsm_open_mac_service
 - Mac, 51
- hsm_open_session
 - Session, 7
- hsm_open_signature_generation_service
 - Signature generation, 31
- hsm_open_signature_verification_service
 - Signature verification, 37
- HSM_OUT_OF_MEMORY
 - Error codes, 75
- HSM_OUT_TOO_SMALL
 - Error codes, 75
- hsm_permitted_algo_t
 - Key management, 18
- hsm_prepare_signature
 - Signature generation, 32
- hsm_pub_key_recovery
 - Public key recovery, 67
- hsm_pubkey_type_t
 - Key management, 17
- HSM_RNG_NOT_STARTED
 - Error codes, 75
- HSM_SELF_TEST_FAILURE
 - Error codes, 75
- hsm_service_hdl_s, 6
- HSM_SERVICES_DISABLED
 - Error codes, 75
- hsm_session_hdl_s, 6
- HSM_SIGNATURE_INVALID
 - Error codes, 75
- hsm_signature_scheme_id_t
 - Signature generation, 30
- hsm_storage_loc_t
 - Key management, 17
- hsm_storage_persist_lv_t
 - Key management, 17
- hsm_svc_cipher_flags_t
 - Ciphering, 24
- hsm_svc_data_storage_flags_t
 - Data storage, 45
- hsm_svc_key_management_flags_t
 - Key management, 17
- HSM_SVC_KEY_STORE_FLAGS_CREATE
 - Key store, 70
- HSM_SVC_KEY_STORE_FLAGS_SET_MAC_LEN
 - Key store, 70
- HSM_SVC_KEY_STORE_FLAGS_STRICT_OPERATION
 - Key store, 70
- hsm_svc_key_store_flags_t
 - Key store, 70
- HSM_UNKNOWN_ERROR
 - Error codes, 75
- HSM_UNKNOWN_HANDLE
 - Error codes, 75
- HSM_UNKNOWN_ID
 - Error codes, 75
- HSM_UNKNOWN_KEY_STORE
 - Error codes, 75
- HSM_UNKNOWN_WARNING
 - Error codes, 75
- HSM_VERIFICATION_STATUS_FAILURE
 - Signature verification, 35
- HSM_VERIFICATION_STATUS_SUCCESS
 - Signature verification, 35
- hsm_verification_status_t
 - Signature verification, 36
- hsm_verify_sign
 - Signature verification, 36
- hsm_verify_signature
 - Signature verification, 37
- kek_enc_key_hdr_t, 13
- Key management, 10
 - FLAG, 16
 - hsm_bit_key_sz_t, 18
 - hsm_close_key_management_service, 20
 - hsm_delete_key, 18
 - hsm_generate_key, 19
 - hsm_generate_key_ext, 19
 - hsm_get_key_attr, 18
 - hsm_import_key, 19
 - hsm_key_group_t, 17
 - HSM_KEY_INFO_KEK, 16
 - HSM_KEY_INFO_MASTER, 16
 - HSM_KEY_INFO_PERMANENT, 16
 - HSM_KEY_INFO_PERSISTENT, 15

- hsm_key_info_t, 17
- HSM_KEY_INFO_TRANSIENT, 16
- hsm_key_lifecycle_t, 18
- hsm_key_lifetime_t, 17
- hsm_key_type_t, 17
- HSM_KEY_USAGE_DECRYPT, 15
- HSM_KEY_USAGE_DERIVE, 15
- HSM_KEY_USAGE_ENCRYPT, 15
- HSM_KEY_USAGE_EXPORT, 15
- HSM_KEY_USAGE_SIGN_HASH, 15
- HSM_KEY_USAGE_SIGN_MSG, 15
- hsm_key_usage_t, 17
- HSM_KEY_USAGE_VERIFY_HASH, 15
- HSM_KEY_USAGE_VERIFY_MSG, 15
- HSM_OP_DEL_KEY_FLAGS_STRICT_OPERATION, 15
- hsm_op_delete_key_flags_t, 16
- hsm_op_import_key_flags_t, 16
- hsm_op_key_gen_flags_t, 17
- HSM_OP_KEY_GENERATION_FLAGS_STRICT_OPERATION, 16
- hsm_open_key_management_service, 20
- hsm_permitted_algo_t, 18
- hsm_pubkey_type_t, 17
- hsm_storage_loc_t, 17
- hsm_storage_persist_lvl_t, 17
- hsm_svc_key_management_flags_t, 17
- Key store, 69
 - hsm_close_key_store_service, 71
 - hsm_open_key_store_service, 70
 - HSM_SVC_KEY_STORE_FLAGS_CREATE, 70
 - HSM_SVC_KEY_STORE_FLAGS_SET_MAC_LEN, 70
 - HSM_SVC_KEY_STORE_FLAGS_STRICT_OPERATION, 70
 - hsm_svc_key_store_flags_t, 70
- Life Cycle update, 72
 - hsm_lc_new_state_t, 72
 - hsm_lc_update, 72
- Mac, 49
 - hsm_close_mac_service, 52
 - hsm_do_mac, 51
 - hsm_mac_one_go, 52
 - HSM_MAC_VERIFICATION_STATUS_SUCCESS, 50
 - hsm_mac_verification_status_t, 50
 - hsm_op_mac_one_go_algo_t, 50
 - HSM_OP_MAC_ONE_GO_FLAGS_MAC_GENERATION, 50
 - HSM_OP_MAC_ONE_GO_FLAGS_MAC_VERIFICATION, 50
 - hsm_op_mac_one_go_flags_t, 50
 - hsm_open_mac_service, 51
- op_auth_enc_args_t, 22
- op_cipher_one_go_args_t, 23
- op_data_storage_args_t, 44
- op_debug_dump_args_t, 54
- op_delete_key_args_t, 12
- op_dev_attest_args_t, 55
- op_dev_getinfo_args_t, 57
- op_gc_acrypto_args_t, 60
- op_gc_akey_gen_args_t, 63
- op_generate_key_args_t, 14
- op_generate_key_ext_args_t, 13
- op_generate_sign_args_t, 29
- op_get_info_args_t, 65
- op_get_key_attr_args_t, 12
- op_get_random_args_t, 39
- op_hash_one_go_args_t, 41
- op_import_key_args_t, 13
- op_lc_update_msg_args_t, 72
- op_mac_one_go_args_t, 49
- op_prepare_sign_args_t, 29
- op_pub_key_recovery_args_t, 67
- op_verify_sign_args_t, 34
- Open key session, 6
- open_svc_cipher_args_t, 23
- open_svc_data_storage_args_t, 44
- open_svc_key_management_args_t, 14
- open_svc_key_store_args_t, 69
- open_svc_mac_args_t, 49
- open_svc_sign_gen_args_t, 29
- open_svc_sign_ver_args_t, 34
- Public key recovery, 67
 - hsm_pub_key_recovery, 67
- Random number generation, 39
 - hsm_do_rng, 39
 - hsm_get_random, 39
- service_hdl_to_ptr
 - Session, 8
- Session, 5
 - add_service, 9
 - add_session, 8
 - delete_service, 8
 - delete_session, 8
 - hsm_close_session, 7
 - hsm_hdl_t, 6
 - hsm_open_session, 7
 - service_hdl_to_ptr, 8
 - session_hdl_to_ptr, 7
- session_hdl_to_ptr
 - Session, 7
- Signature generation, 28
 - hsm_close_signature_generation_service, 31
 - hsm_do_sign, 30
 - hsm_generate_signature, 31
 - HSM_OP_GENERATE_SIGN_FLAGS_INPUT_DIGEST, 29
 - HSM_OP_GENERATE_SIGN_FLAGS_INPUT_MESSAGE, 29
 - hsm_op_generate_sign_flags_t, 30
 - hsm_op_prepare_signature_flags_t, 30

- hsm_open_signature_generation_service, [31](#)
- hsm_prepare_signature, [32](#)
- hsm_signature_scheme_id_t, [30](#)
- Signature verification, [34](#)
 - hsm_close_signature_verification_service, [37](#)
 - HSM_OP_VERIFY_SIGN_FLAGS_COMPRESSED_POINT,
[35](#)
 - HSM_OP_VERIFY_SIGN_FLAGS_INPUT_DIGEST,
[35](#)
 - HSM_OP_VERIFY_SIGN_FLAGS_INPUT_MESSAGE,
[35](#)
 - HSM_OP_VERIFY_SIGN_FLAGS_KEY_INTERNAL,
[35](#)
 - hsm_op_verify_sign_flags_t, [36](#)
 - hsm_open_signature_verification_service, [37](#)
 - HSM_VERIFICATION_STATUS_FAILURE, [35](#)
 - HSM_VERIFICATION_STATUS_SUCCESS, [35](#)
 - hsm_verification_status_t, [36](#)
 - hsm_verify_sign, [36](#)
 - hsm_verify_signature, [37](#)